ARMY RESEARCH LABORATORY

Meteorological Sensor Array (MSA)–Phase I, Volume 2 (Data Management Tool: "Proof of Concept")

by Sandra Harrison and Gail Vaucher

ARL-TR-7133 **October 2014**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
White Sands Missile Range, NM 88002-5501

ARL-TR-7133 **October 2014**

Meteorological Sensor Array (MSA)–Phase I, Volume 2 (Data Management Tool: "Proof of Concept")

Sandra Harrison
STS Systems Integration (SSI), LLC
San Antonio, TX

Gail Vaucher
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
October 2014	Final	March 2014–September 30, 2014

4. TITLE AND SUBTITLE
Meteorological Sensor Array (MSA)–Phase I, Volume 2 (Data Management Tool: "Proof of Concept")

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Sandra Harrison and Gail Vaucher

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
Computational and Information Sciences Directorate
Battlefield Environment Division (ATTN: RDRL-CIE-D)
White Sands Missile Range, NM 88002-5501

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-7133

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The Meteorological Sensor Array (MSA) goal is to provide reliable and persistent atmospheric data resources, which allow atmospheric modelers and sensor developers to validate and compare model and sensor performance with observations at and near the surface and in close proximity to terrain of varying complexity. The MSA–Phase I ("Proof of Concept") field campaign was executed in 2014. Concurrently, a "Proof of Concept" data management tool was designed, created and initial tests conducted. The Data Management task was divided into 2 parts: Data Processing and Data Distribution. The Data Processing was defined as the data flow from field sensor measurements through the initial data averaging, data merging, and time-series visualization plots used to quality control the data. The Data Distribution task began once the field data were quality controlled and included the process of reconfiguring the field data files for MSA data users. An earlier report (ARL-TR-7058) describes the Data Processing, in detail. Volume 2 focuses on the subsequent 2014-Data Distribution Tool development. This Data Distribution Tool consisted of 4 functions: Data Storage, Data Extraction, Data Visualization, and Data Documentation. Each function is elaborated in this report, along with a sample of the "lessons learned". Data management is a nontrivial, critical pillar of the MSA. With software technologies advancing daily, the possibilities for improved data management tools are almost open ended. With this documentation, however, a foundation for creating a successful MSA Data Management Tool has been initiated.

15. SUBJECT TERMS
meteorological sensor array, MSA, data management tool, data processing, data distribution, Access, MySQL

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Gail Vaucher
Unclassified	Unclassified	Unclassified	UU	82	19b. TELEPHONE NUMBER (*Include area code*) 575-678-3237

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

INTENTIONALLY LEFT BLANK.

Acknowledgments

The authors wish to thank Mr “RW” Hornbaker (SpecPro Technical Services [STS] Systems Integration, [SSI], LLC) and Mr. Pieter Haines (STS), for their invaluable contributions toward the development and testing of the Meteorological Sensor Array (MSA)–Phase I: Data Management Tool.

Special thanks go to the Technical Publishing Branch at White Sands Missile Range, NM, for their consistent high standard of editing, specifically Mr Jim Le Noir, Ms Lisa Lacey, and Sherry Larson.

INTENTIONALLY LEFT BLANK.

Executive Summary

Accurate atmospheric models are a critical element of environmentally-dependent Army decisions. “Army-scale” atmospheric models include high-resolution (≤1-km) models. To validate these models, high-resolution meteorological observations are needed. Locating atmospheric measurements at scales of 1 km or less is very difficult. This fact was recognized by the National Research Council (NRC). The US Army Research Laboratory (ARL) has responded to this critical, national and military technological gap by proposing to build an observational data resource specifically designed to address the “Army-scale”, high-resolution atmospheric model validation and verification issues. The resource is called the Meteorological Sensor Array (MSA).

The MSA objective is to provide reliable and persistent atmospheric data resources, which allow atmospheric modelers and sensor developers to validate and compare model and sensor performance with observations at and near the surface and in close proximity to terrain of varying complexity. The multiphase MSA program was initiated in 2014, with a MSA–Phase I “Proof of Concept”. The field portion of Phase I consisted of the following: a) equally-spaced meteorological towers located around a large Solar Photovoltaic Farm in southern New Mexico; b) measurements of pressure, temperature, relative humidity, insolation, and winds; c) solar-powered instrumentation; and d) wireless data download, monitoring, and time synchronization. The subsequent phases were envisioned as having 36 meteorological towers placed in a gridded pattern at multiple desert locations in the southwestern USA.

The MSA–Phase I field design and execution were documented in MSA–Phase I, Volume 1.[1] The MSA–Phase I Data Management task was divided into 2 parts: Data Processing and Data Distribution. The Data Processing was defined as the data flow from measurements sampled by a sensor in the field, through the initial data averaging, data merging, and time-series visualization plots used to execute the initial data quality control, at the MSA Headquarters. MSA–Phase I, Volume 1[1] described this part, in detail. The Data Distribution task was defined as starting after the MSA field data were quality controlled and as being the process of reconfiguring the field data files for the MSA end users. The tool developed to execute the Data Distribution is the primary focus of this technical report.

The MSA–Phase I Data Distribution Tool consists of 4 functions: Data Storage, Data Extraction, Data Visualization, and Data Documentation.

[1]Vaucher GT, Swanson J, Raby J, Foley T, Harrison S, Brice R. D’Arcy S, Creegan E. Meteorological Sensor Array (MSA)–Phase I, Volume 1 (“Proof of Concept” Overview). White Sands Missile Range (NM): Army Research Laboratory (US); September 2014. Report No.: ARL-TR-7058. Also available at http://www.arl.army.mil/www/default.cfm?technical_report=7179.

The Phase I: Data Storage was designed as a Windows MySQL database. The development of this Phase I: Data Storage function, however, was subdivided into 3 steps. Due to delays in the software acquisition, the initial data storage tool utilized the available Microsoft Access[2] database. Figure 1 shows the MSA–Phase I database table design and relationships. A Visual Basic Application script (Appendix A) was used in Microsoft Access to read and manage all the MSA Merged Data files.

As an intermediate development step, the "Proof of Concept" data storage was reinstalled in a UNIX-based MySQL environment. Populating the database was done using a Visual Basic Application script (Appendix B) to read all the MSA Merged Data files and write them into a formatted text file for each database table. A File Transfer Protocol (FTP) was used to transfer these text files to the UNIX workstation. The MySQL[3] LOAD command was then used to load the tables with these data in text files. Throughout the development process of the Data Management Tool, there were database table improvements. Implementing the improvements required recreating the tables and reloading the data into the tables. To improve efficiency, this process was automated.

The final step converted UNIX MySQL to Windows-based MySQL. At the time of this writing, the conversion step was in progress.

Looking forward, the "Proof of Concept" data storage specifications were used to assess current and future database size requirements. Based on the Phase I data volume, in 1 year (yr), for a single tower, the maximum amount of disk storage needed for a merged data table would be approximately 128 megabytes (MB). For 36 towers, the merged data table size would be approximately 5-GB per year. For 72 towers, approximately 10-GB per year would be required.

The subsequent 3 Data Distribution Tool functions were initiated by the MSA Operator, using desktop icons. To help visualize these 3 functions, here is an example of their usage from an operator's perspective:

> When a user submitted a data request, the MSA Operator began processing this request by clicking on the Data Management Extract function. The function initiated PuTTY software, a terminal emulator providing access to the UNIX environment required by MySQL database. The function then executed the extraction program *MSA Extract*. The Operator entered the requested specifications, which then extracted the MSA–Phase I data stored in a MySQL database. There was an option for choosing "all" variables, after which one could reduce the selection through a toggle interface. The data requested were instantaneously processed, and a sample of the results was shown on the screen. A Data Management FTP icon facilitated a UNIX-to-Windows data file transfer. The extracted

[2]Microsoft Access is a registered trademark of Microsoft Corporation in the United States and/or other countries.

[3]The SQL part of "MySQL" stands for Structured Query Language.

numerical file was converted into color-coded time-series plots, using the Data Management Graphics Layout Engine (GLE) Plot icon. Clicking on the final icon—Data Management Excel Log and using local key strokes, allowed the Operator to tabulate the data request into an historical record of all MSA data requests received and processed, to date.

This report elaborates on each of the 4 Data Distribution Tool functions in Section 2.

Lessons learned were gleaned throughout the MSA–Phase I Data Management "Proof of Concept" exercise. Four of these lessons included:

1) The process of acquiring needed Software Application Tools takes a significant amount of time.

2) Requesting multiple towers with 1 query for 1 output data file was very complex. This task was simplified by having separate queries and output files generated for each tower.

3) A Rapid Application Development Methodology was used, based on time constraints and the availability of fundamental algorithms needed to build the database queries. This application used a command line interface, which does not easily accommodate user entry errors. Using a graphical user interface (GUI) could be more user-friendly.

4) While the Data Management Tool successfully proved the initial data management concept, utilizing a Web-based application would be recommended for the next development stage(s). A Web-based application would have GUI capabilities (reducing the command line interface limitations) and the potential for reaching more MSA users.

Data Management is a nontrivial, critical pillar of a MSA. With software technologies advancing daily, the possibilities for improved user-friendly, efficient, and informative tools are almost open ended. With this documentation, a foundation for creating a successful MSA Data Management Tool has been initiated.

INTENTIONALLY LEFT BLANK.

1. Background

Environmentally dependent Army decisions rely on accurate atmospheric models. “Army-scale” atmospheric models include high-resolution (≤1-km) models. To validate these models, high-resolution meteorological observations are needed. As explained in Meteorological Sensor Array (MSA)–Phase I, Volume 1,[1] locating atmospheric measurements at scales of 1 km or less, is very difficult. This fact was recognized by the National Research Council (NRC), after they reviewed the US Weather Research, and Researcher-to-Operations progress and priorities in 2009.[2] The US Army Research Laboratory (ARL) has responded to this critical, national and military technological gap by proposing to build an observational data resource specifically designed to address the “Army-scale”, high-resolution atmospheric model validation and verification issues. The resource is called the Meteorological Sensor Array (MSA).

1.1 MSA Vision

The MSA vision is to provide reliable and persistent atmospheric data resources, which allow atmospheric modelers and sensor developers to validate and compare model and sensor performance with observations at and near the surface and in close proximity to terrain of varying complexity. The multiphased MSA program was initiated in 2014, with a MSA–Phase I “Proof of Concept”. The field portion of Phase I consisted of:

- Five equally-spaced meteorological towers located around a large Solar Photovoltaic (PV) Farm in southern New Mexico
- Measurements of pressure, temperature, relative humidity, insolation and winds
- Solar-powered instrumentation
- Wireless data download, monitoring, and time synchronization.

The subsequent phases were envisioned as having 36 meteorological towers placed in a gridded pattern at multiple desert locations in the southwestern USA. Supplemental volume measurements, such as triple LIDARS, were included in the evolving middle 2 phases. Later MSA Phases were visualized as being a mobile measurement capability, which will be designed for integration into other remote site field campaigns. For this document, however, the focus is on MSA–Phase I.

1.2 MSA–Phase I (“Proof of Concept”) Field Campaign

The MSA–Phase I (“Proof of Concept”) field measurements were sampled from portable lightweight aluminum towers, with sensors mounted at the 2- and 10-m above ground level (AGL). The data acquisition systems (DASs) were divided into 2 categories: Thermodynamic and Dynamic DAS. A Campbell Scientific CR23X micrologger assimilated the Thermodynamic

1-min averaged data. The variables consisted of pressure, temperature (2- and 10-m AGL), relative humidity (2-m AGL), and insolation (2-m AGL). The Dynamic data originated from 2 RM Young 81000 Ultrasonic anemometers (2- and 10-m AGL) sampling at 20 Hz. The variables acquired were wind speed, wind direction, u-component, v-component, w-component, speed of sound and sonic-temperature. The raw dynamic data were preserved in files on a laptop computer then reduced to 1-min averages. These 1-min averages were then merged with the thermodynamic 1-min data. A multiport adapter bridged the 2 data resources. A system clock on each tower was synchronized using the Network Time Protocol (NTP). For a more detailed MSA–Phase I description, see MSA–Phase I, Volume 1.[1]

2. MSA Data Management Tool, Design, and Process

The MSA–Phase I Data Management task was divided into 2 parts: Data Processing and Data Distribution. The Data Processing was defined as the data flow from measurements sampled by a sensor in the field, through the initial data averaging, data merging, and time-series visualization plots used to execute the initial data quality control (QC), at the MSA Headquarters. This Data Processing portion produced time-series plots that were critical to the Phase I Field Campaign's daily data reviews. A detailed data processing description is in ARL-TR-7058.[1]

The Data Distribution subtask started after the field data were quality controlled. The subtask's underlying objective was to reconfigure the field data files for the MSA end users. A preview sample of this reconfiguration was included in Volume 1,[1] where the "data processing" description included the task of reformatting the data to accommodate a user's statistical application. In that case, testing the feasibility of model validation and verification (V&V) with MSA observational data had been selected as the MSA data application. A subset of the MSA–Phase I data was selected and reformatted into Model Evaluation Tool (MET) American Standard Code for Information Interchange (ASCII) input, for validating the Weather Running Estimate-Nowcast (WRE-N) model.[3] The success of the feasibility test laid the groundwork for developing a full Data Management–Data Distribution Tool, which is the focus of the remaining technical report.

Note: While updated fiscal year (FY) 2015 requirements continue to evolve the Data Distribution Tool, this report documents the significant FY14, MSA–Phase I "Proof of Concept" Data Distribution milestones.

2.1 Data Distribution Tool Overview

The MSA–Phase I Data Distribution Tool consists of 4 functions: Data Storage, Data Extraction, Data Visualizations, and Data Documentation. The data storage will be explained in Section 2.2.

The subsequent functions were initiated by the MSA Operator, using desktop icons. To help visualize these 3 functions, a description of their usage from an operator's perspective is provided:

> When a user submitted a data request, the MSA Operator began processing this request by clicking on the Data Management Extract function. The function initiated PuTTY software, a terminal emulator providing access to the UNIX environment required by "My Structured Query Language" (MySQL) database. The function then executed the extraction program *MSA Extract*. The Operator then entered their initials, followed by the user's initials. The requested data's local date and time were entered, followed by the tower and variable selections. There was an option for choosing "all" variables, after which one could reduce the selection through a toggle interface. The data requested were instantaneously extracted from the MySQL database, and a sample of the results was shown on the screen. A Data Management File Transfer Protocol (FTP) icon facilitated a UNIX-to-Windows data file transfer. The extracted numerical file was converted into color-coded time-series plots, using the Data Management Graphics Layout Engine (GLE) Plot icon. Clicking on the final icon—Data Management Excel Log and using local key strokes, allowed the operator to tabulate the data request into an historical record of all MSA data requests received and processed, to date.

An elaboration of the Data Distribution Tool's 4 functions will be given in the next 4 sections, starting with the data storage.

2.2 Data Storage

The Phase I-Data Storage was designed to use a MySQL database. Due to delays in the software acquisition, the initial data storage tool utilized the available Microsoft Access[4] database. Database tables and their relationships were defined, using the Access relationship editor and tools. A decision was made to store all the merged data in 1 table. This choice eliminated the overhead of "join" statements, which would be needed if a multiple-table design was used to house the merged data. A second table was created to store the header information of each data file. This table had a "one-to-many" relationship with the merged data table. That is, for each row in the header table, there existed many rows in the merged data table. A third table was created for the towers and their attributes. This table was also configured in a "one-to-many" relationship from the tower table to the merged data table; there existed many rows in the merged data table for each row in the Tower table. Figure 1 shows the MSA–Phase I table design and relationships.

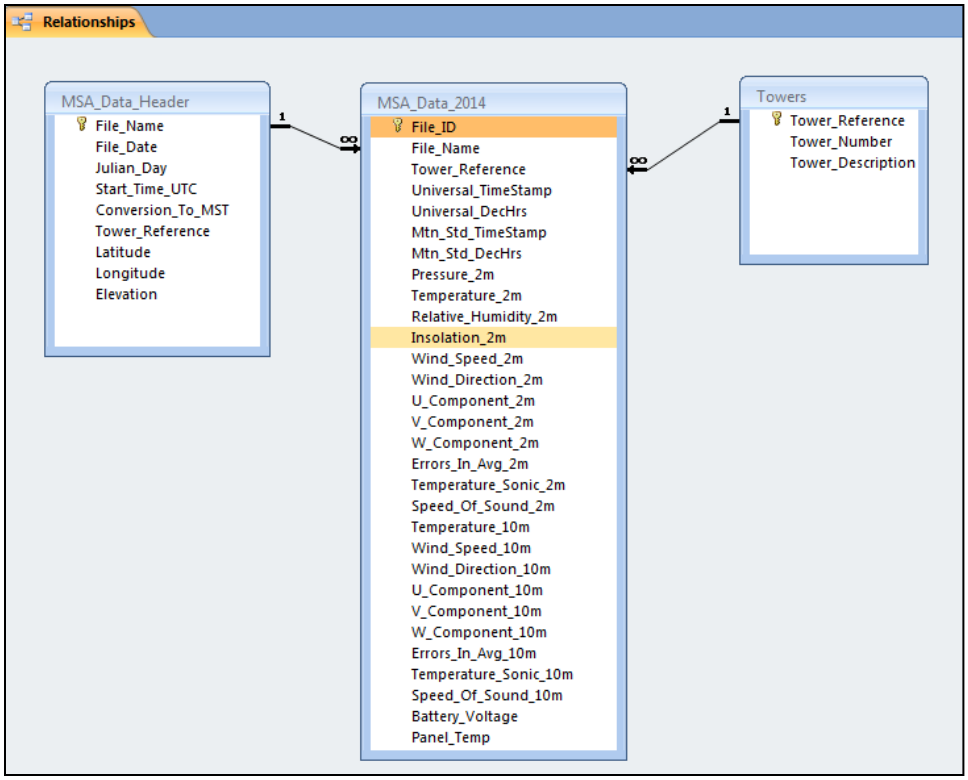


Fig. 1 MSA–Phase I database design including tables and relationships

A Visual Basic Application script was used in Access to read all the MSA Merged Data files from the given start and end dates (see Appendix A). This script loaded the data into the Access tables, and queries were run to test the efficiency of the database design. If a problem occurred with a table or relationship, the delay or abnormality would be reflected in the query's efficiency report.

Once the initial software acquisition hurdles were passed, the "Proof of Concept" data storage was reinstalled in a UNIX-based MySQL environment. The reinstallation was an intermediate step toward creating the intended Windows MySQL database. A Visual Basic Application script was written that read all the MSA Merged Data files from the given start and end dates, and then wrote all these data into a formatted text file for each database table (Appendix B). An FTP was used to transfer these text files to the UNIX workstation. The MySQL LOAD command was then used to load the tables with these data in text files. Throughout the development process of the Data Management Tool Application, there were some changes that needed to be made to the database tables. The process included recreating the tables and reloading the data into the tables from the text files. To improve efficiency, this process was automated.

Looking to the future, the "Proof of Concept" data storage specifications were used to assess current and future database size requirements. For example, based on the Phase I data volume, in 1 yr, for a single tower, the maximum amount of disk storage needed for a merged data table would be approximately 128 megabytes (MB). This estimate was based on storing 1-min

averaged data. As shown in the Table, storing 1-min averages meant that the data rows per year equaled the minutes per year. A single tower data sample used 30 fields. Each field required 8-bytes per field. Thus, over a 1-yr period, 128 MB would be required.

Table Calculating a MSA database table size, for a single tower, over a 1-yr period

60 min/h * 24 h/day * 365 days/yr = 525,600 min/yr (data rows per year)
For 1 Tower: 30 fields * 8 Bytes/field = 240 Bytes per row
240 Bytes/row * 525,600 rows/yr equals ~128 MB/yr, for 1 tower

For 36 towers, the merged data table size would be approximately 5-GB per year. For 72 towers, approximately 10-GB per year would be required.

2.3 Data Extraction

The Database extraction process was implemented using a UNIX shell script language. This application featured a command line interface that prompted the user for input specifications (See Fig. 2).

User input included the:

1) MSA operator initials
2) Data Requestor initials
3) Data Start and End Dates (in Mountain Standard Time [MST])
4) Data Start and End Times (in MST)
5) Towers selected
6) Variables selected (i.e., Pressure, Wind Speed, U-component, V-component, W-Components, etc.)

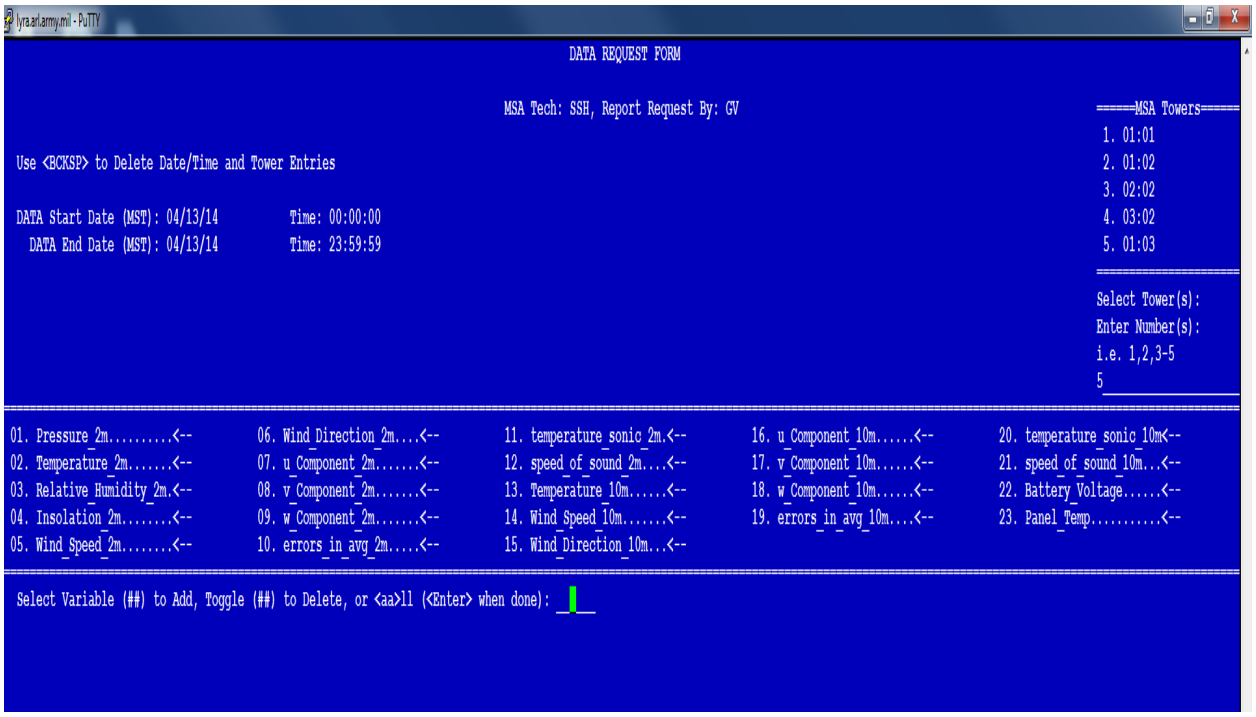


Fig. 2 An example of the Data Management Tool Application user inputs in the database extraction process

Based on the user's input data, the program assembled a MySQL database select statement or query, and executed the command in the database. The returned data were captured in a text file on disk—1 text file for each tower requested. The text file had a descriptive name that included all the input data information (i.e., Start and End Date/Time, Tower[s], Parameter[s]).

The program also displayed the data on the screen in column format (Fig. 3a) and then displayed the query select statement that generated the data extraction below the data display (Fig. 3b). Following the query display, a message was shown instructing the user to FTP and plot the data (Fig. 3c).

lyra.arl.army.mil - PuTTY

Mtn_Std_TimeStamp	Mtn_Std_DecHrs	Tower_Reference	u_2m	v_2m	w_2m	u_10m	v_10m	w_10m
20140413 000000	0	01:01	5.03	-1.36	-0.32	7.4	-1.04	-0.14
20140413 000100	0.017	01:01	5.81	-1.66	-0.3	9.37	-1.31	-0.45
20140413 000200	0.033	01:01	7.68	-0.52	-0.51	11.15	0.54	-0.53
20140413 000300	0.05	01:01	5.57	0.97	-0.34	8.49	2.24	-0.43
20140413 000400	0.067	01:01	6.62	-0.15	-0.32	9.28	1.01	-0.33
20140413 000500	0.083	01:01	6.03	-0.68	-0.22	9.4	-0.6	-0.43
20140413 000600	0.1	01:01	6.77	-1.68	-0.29	11.12	-1.58	-0.38
20140413 000700	0.117	01:01	5.61	0.16	-0.27	8.57	2.05	-0.02
20140413 000800	0.133	01:01	4.96	0.59	-0.2	8	0.97	-0.37
20140413 000900	0.15	01:01	5.73	-0.82	-0.22	8.81	-0.69	-0.35
20140413 001000	0.167	01:01	5.28	-1.34	-0.32	8.54	-1.6	-0.49
20140413 001100	0.183	01:01	6.13	-1.53	-0.43	9.32	-1.7	-0.58
20140413 001200	0.2	01:01	6.63	-1.21	-0.35	10.48	-1.26	-0.48
20140413 001300	0.217	01:01	5.81	-1.3	-0.22	9.87	-1.69	-0.33
20140413 001400	0.233	01:01	6.78	-1.21	-0.42	10.35	0.01	-0.69
20140413 001500	0.25	01:01	6.52	-1.61	-0.32	9.76	-2.17	-0.64
20140413 001600	0.267	01:01	6.38	-0.45	-0.41	8.8	0.41	-0.13
20140413 001700	0.283	01:01	6.75	-1.73	-0.24	11.18	-1.74	-0.48
20140413 001800	0.3	01:01	7.56	-2.99	-0.49	12.77	-4.03	-0.53
20140413 001900	0.317	01:01	7.79	-0.99	-0.28	12.63	-0.42	-0.46
20140413 002000	0.333	01:01	6.69	0.48	-0.46	10	2.09	-0.49
20140413 002100	0.35	01:01	5.12	0.28	-0.29	7.91	1.63	-0.02
20140413 002200	0.367	01:01	6.09	-0.28	-0.4	9.21	0.4	-0.24
20140413 002300	0.383	01:01	8.06	-2.17	-0.52	11.89	-2.08	-0.61
20140413 002400	0.4	01:01	6.76	-0.78	-0.47	9.55	-0.98	-0.41
20140413 002500	0.417	01:01	6.49	-2.19	-0.37	10.18	-2.77	-0.56
20140413 002600	0.433	01:01	5.31	-3.93	-0.43	8.57	-5.63	-0.12
20140413 002700	0.45	01:01	5.76	-1.87	-0.38	8.45	-2.42	-0.17
20140413 002800	0.467	01:01	5.52	-1.41	-0.39	7.87	-1.38	-0.35
20140413 002900	0.483	01:01	4.49	-1.82	-0.22	7.46	-2.99	-0.14
20140413 003000	0.5	01:01	6.01	-1.49	-0.44	8.64	-0.72	-0.63
20140413 003100	0.517	01:01	5.75	-0.71	-0.31	8.42	-0.18	-0.17
20140413 003200	0.533	01:01	6.64	-2.61	-0.33	10.92	-3.47	-0.55
20140413 003300	0.55	01:01	6.49	-3.73	-0.48	10.48	-5.09	-0.32
20140413 003400	0.567	01:01	7.71	-1.91	-0.51	11.44	-1.45	-0.44
20140413 003500	0.583	01:01	6.81	0.21	-0.43	9.84	0.97	-0.31
20140413 003600	0.6	01:01	5.95	-0.37	-0.35	9.29	-0.03	-0.27
20140413 003700	0.617	01:01	6.07	-0.46	-0.33	9.42	0.05	-0.26

a)

```
SELECT Mtn_Std_TimeStamp, Mtn_Std_DecHrs, Tower_Reference, u_2m, v_2m, w_2m, u_10m, v_10m, w_10m
FROM test.MSA_DATA
WHERE ((Mtn_Std_TimeStamp BETWEEN '2014-04-13 00:00:00' AND '2014-04-13 23:59:59') AND (Tower_Reference='01:01'));
```

b)

```
1) TRANSFER
   /home/sandrah/Reports/gv/M_1404130000_1404132359_11_2uvw_10uvw_  and MSAaccess.txt from Unix:Lyra
   to Windows:MSA C:\MSA\MSA_SH\DM_Tools  Using FileZilla
2) PLOT Data in GLE with MSA_Plots_MST.vbs and view LOG FILE (C:\...\Logs\MSAaccess.txt)  with Excel
```

c)

Fig. 3 A Data Management Tool example screen display, after the user validated the input. This output included a) extracted data in tabular format, b) a MySQL Select statement created from the input data and sent to the database to generate the data, and c) the remaining procedures to perform for producing the final data products.

Extracted data (Fig. 4) and Log files were generated and transferred from the UNIX workstation to the MSA–Phase I Windows workstation. A GLE script was used to plot the data (see Fig. 5) and view the log in Excel (see Fig. 6).

```
!PTRISDuvwexc_TSDuvwexc,BP
!Header Info
20140413 00000 0.00 01:03 -99.99 20.26 15.41 0.00 4.67 284.90 4.51 -1.20 -0.42 0 18.91 343.07 20.29 7.45 286.80 7.13 -2.15 -0.32 0 17.09 341.99 12.31 21.65
20140413 00100 0.01 01:03 -99.99 20.16 15.73 0.00 6.03 275.80 5.99 -0.61 -0.40 0 18.86 343.03 20.25 9.77 276.40 9.70 -1.09 -0.47 0 17.00 341.94 12.32 21.64
20140413 00200 0.03 01:03 -99.99 20.15 15.45 0.00 6.29 261.50 6.22 0.93 -0.48 0 18.89 343.05 20.21 9.37 266.70 9.36 0.55 -0.46 0 17.04 341.97 12.31 21.63
20140413 00300 0.05 01:03 -99.99 20.05 15.23 0.00 4.85 259.90 4.77 0.85 -0.44 0 18.85 343.03 20.11 7.58 262.40 7.51 1.01 -0.23 0 17.00 341.94 12.32 21.62
20140413 00400 0.06 01:03 -99.99 20.07 15.28 0.00 4.51 274.20 4.49 -0.33 -0.21 0 18.73 342.96 20.11 7.71 272.60 7.70 -0.35 -0.19 0 16.93 341.90 12.31 21.61
20140413 00500 0.08 01:03 -99.99 20.03 15.33 0.00 5.56 266.30 5.55 0.36 -0.52 0 18.76 342.98 20.07 9.11 269.50 9.11 0.08 -0.40 0 16.96 341.92 12.31 21.60
20140413 00600 0.10 01:03 -99.99 19.94 15.57 0.00 6.89 276.40 6.85 -0.77 -0.51 0 19.03 343.14 20.00 11.23 276.30 11.16 -1.22 -0.54 0 17.21 342.06 12.31 21.59
20140413 00700 0.11 01:03 -99.99 19.86 15.51 0.00 5.78 262.20 5.73 0.79 -0.41 0 18.98 343.11 19.95 8.89 256.40 8.64 2.10 -0.42 0 17.15 342.03 12.31 21.59
20140413 00800 0.13 01:03 -99.99 20.00 15.04 0.00 4.16 252.90 3.98 1.23 -0.25 0 18.90 343.06 20.18 7.76 253.10 7.43 2.25 -0.06 0 17.20 342.06 12.31 21.58
20140413 00900 0.15 01:03 -99.99 20.15 14.89 0.00 6.27 260.10 6.18 1.08 -0.42 0 18.95 343.09 20.21 10.53 267.70 10.52 0.42 -0.58 0 17.11 342.00 12.31 21.56
20140413 01000 0.16 01:03 -99.99 20.09 14.58 0.00 5.77 270.10 5.77 -0.01 -0.61 0 18.79 342.99 20.19 9.17 277.30 9.09 -1.16 -0.64 0 16.94 341.91 12.31 21.55
20140413 01100 0.18 01:03 -99.99 20.09 14.88 0.00 6.59 273.50 6.57 -0.40 -0.52 0 18.81 343.01 20.20 10.60 275.80 10.54 -1.07 -0.49 0 17.04 341.97 12.31 21.55
20140413 01200 0.20 01:03 -99.99 20.02 15.32 0.00 6.31 265.80 6.29 0.46 -0.62 0 18.99 343.11 20.07 9.30 266.80 9.28 0.52 -0.61 0 17.11 342.00 12.31 21.53
20140413 01300 0.21 01:03 -99.99 19.95 14.48 0.00 5.62 265.90 5.60 0.40 -0.38 0 18.80 343.00 20.06 9.60 271.20 9.60 -0.20 -0.60 0 17.00 341.94 12.31 21.52
20140413 01400 0.23 01:03 -99.99 20.10 13.91 0.00 5.19 259.90 5.11 0.91 -0.22 0 18.81 343.00 20.20 9.49 258.80 9.31 1.84 -0.19 0 17.06 341.97 12.31 21.51
20140413 01500 0.25 01:03 -99.99 20.08 14.22 0.00 6.81 270.80 6.81 -0.10 -0.53 0 18.80 343.00 20.11 10.64 270.70 10.64 -0.14 -0.64 0 16.96 341.92 12.31 21.50
```

Fig. 4 An example of a data file extracted from the Data Management Tool database

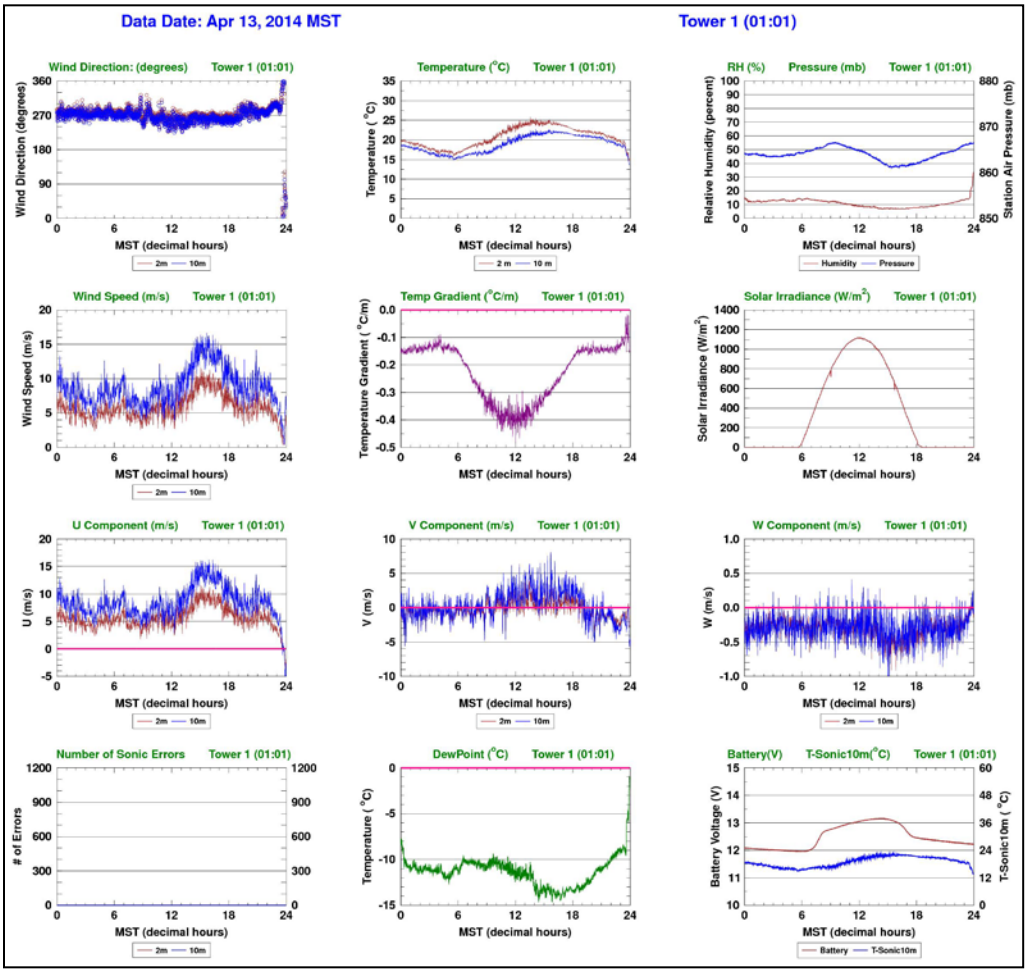
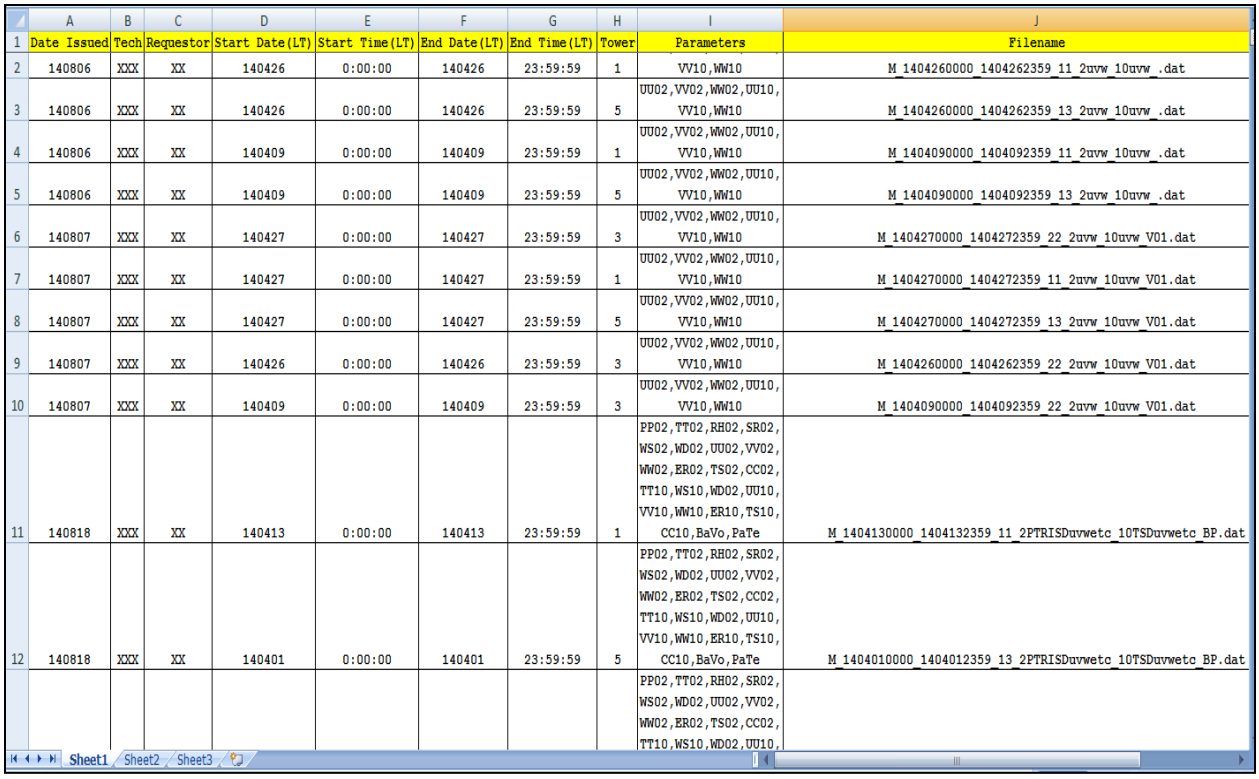


Fig. 5 A GLE script created time-series plots of extracted data



	A	B	C	D	E	F	G	H	I	J
1	Date Issued	Tech	Requestor	Start Date(LT)	Start Time(LT)	End Date(LT)	End Time(LT)	Tower	Parameters	Filename
2	140806	XXX	XX	140426	0:00:00	140426	23:59:59	1	VV10,WW10	M_1404260000_1404262359_11_2uvw_10uvw_.dat
3	140806	XXX	XX	140426	0:00:00	140426	23:59:59	5	UU02,VV02,WW02,UU10, VV10,WW10	M_1404260000_1404262359_13_2uvw_10uvw_.dat
4	140806	XXX	XX	140409	0:00:00	140409	23:59:59	1	UU02,VV02,WW02,UU10, VV10,WW10	M_1404090000_1404092359_11_2uvw_10uvw_.dat
5	140806	XXX	XX	140409	0:00:00	140409	23:59:59	5	UU02,VV02,WW02,UU10, VV10,WW10	M_1404090000_1404092359_13_2uvw_10uvw_.dat
6	140807	XXX	XX	140427	0:00:00	140427	23:59:59	3	UU02,VV02,WW02,UU10, VV10,WW10	M_1404270000_1404272359_22_2uvw_10uvw_V01.dat
7	140807	XXX	XX	140427	0:00:00	140427	23:59:59	1	UU02,VV02,WW02,UU10, VV10,WW10	M_1404270000_1404272359_11_2uvw_10uvw_V01.dat
8	140807	XXX	XX	140427	0:00:00	140427	23:59:59	5	UU02,VV02,WW02,UU10, VV10,WW10	M_1404270000_1404272359_13_2uvw_10uvw_V01.dat
9	140807	XXX	XX	140426	0:00:00	140426	23:59:59	3	UU02,VV02,WW02,UU10, VV10,WW10	M_1404260000_1404262359_22_2uvw_10uvw_V01.dat
10	140807	XXX	XX	140409	0:00:00	140409	23:59:59	3	UU02,VV02,WW02,UU10, VV10,WW10	M_1404090000_1404092359_22_2uvw_10uvw_V01.dat
11	140818	XXX	XX	140413	0:00:00	140413	23:59:59	1	PP02,TT02,RH02,SR02, WS02,WD02,UU02,VV02, WW02,ER02,TS02,CC02, TT10,WS10,WD02,UU10, VV10,WW10,ER10,TS10, CC10,BaVo,PaTe	M_1404130000_1404132359_11_2PTRISDuvwetc_10TSDuvwetc_BP.dat
12	140818	XXX	XX	140401	0:00:00	140401	23:59:59	5	PP02,TT02,RH02,SR02, WS02,WD02,UU02,VV02, WW02,ER02,TS02,CC02, TT10,WS10,WD02,UU10, VV10,WW10,ER10,TS10, CC10,BaVo,PaTe	M_1404010000_1404012359_13_2PTRISDuvwetc_10TSDuvwetc_BP.dat
									PP02,TT02,RH02,SR02, WS02,WD02,UU02,VV02, WW02,ER02,TS02,CC02, TT10,WS10,WD02,UU10,	

Fig. 6 An example of the Log file, listing previous data requests, viewed in Excel

2.4 Data Visualization

GLE Data Visualization Plots were created from the text data files. A Visual Basic Script MSA_Plots_MST.vbs (Appendix C) was invoked that prompted the user for the data filename. The Script then extracted the parameter information from the descriptive data filename and wrote the parameter string to the first line of the text data file for GLE to parse. The Visual Basic Script called GLE script “MSA_Plots_MST.GLE” (Appendix D) that decoded the parameter string from the text data file, to define what data and plots to display. The plots and the text data files were the 2 products delivered to the end user.

2.5 Data Documentation

A HELP file and README file were accessible from the application and on disk. The README file gave a detailed explanation of the descriptive file name of the output data file (Appendix E). The HELP file gave a detailed explanation of how to use the Data Management Tool application, and the different options available (Appendix F). Regarding the application usage, a description and format for each of the input parameters were given, along with a description on how to recover from a user input error. Regarding Data Management Tool options, these included functions such as the “Freeze” and “Log” options. For the user that needed to run the data extraction a number of times, a “Freeze” option allowed the user to set or freeze one or more of the input values that would not change for subsequent runs. The “Log”

option is a formatted screen view of previous queries made by the requestor. As the Data Management Tool evolves, additional options will be added to the HELP file.

3. Lessons Learned and Recommendations

Lessons learned were gleaned throughout the MSA–Phase I Data Management "Proof of Concept" exercise. In this section, we flag 4 lessons that may prove constructive toward the future MSA Phases:

1) Acquiring and installing the Software Application Tools needed to develop the Data Management Tool was challenging. The intricacies of the Software Request Procedure generated several delays.

2) Requesting multiple towers with 1 query for 1 output data file was a very complex task. The task was resolved by implementing multiple towers as a separate query and creating output data files for each tower requested. This solution lent itself to the latest GLE user plots, whose parameters are represented on one page, for each tower.

3) A Rapid Application Development Methodology was used, based on the time constraints and the availability of the fundamental algorithms needed to build the Database queries. The application, however, was limited by the command line interface. For example, in a command line interface, once the user validated the input values (i.e., Date/Time, Tower Reference, Parameters), the user could not change this value without quitting the program and starting again. One solution is to use a graphical user interface (GUI). In a GUI, these limitations do not usually exist. The user can change any input value, any number of times, and only 1 validation is required before the query is assembled and executed.

4) The Data Management Tool was intended as a "Proof of Concept". For the next development effort, we recommend and have begun investigating, a Web-based application. A Web-based application has GUI capabilities (reducing command line interface limitations) and the potential for reaching more MSA users.

The current Data Management Tool has successfully extracted user-requested data from database storage, graphically displayed the variables in times series, and logged the data distribution. One of the next steps is to create detailed measurement site and data format descriptions to accompany the extracted data. Creating these descriptive documents will be addressed in future reports.

4. Summary

The MSA–Phase I ("Proof of Concept") purpose, vision and field execution were summarized in Section 1. In short, the goal for the multiphased MSA program is to provide reliable and persistent atmospheric data resources, which allow atmospheric modelers and sensor developers to validate and compare model and sensor performance with observations at and near the surface and in close proximity to terrain of varying complexity. The MSA–Phase I field campaign was executed in 2014. As part of this field event, a "Proof of Concept" data management tool was designed, created, and initial tests conducted.

Data Management is a robust, nontrivial pillar in the MSA program. To reduce the Data Management task into a more manageable assignment, the "Proof of Concept" version was divided into 2 parts: Data Processing and Data Distribution. The Data Processing was defined as the data flow from measurements sampled by a sensor in the field, through the initial data averaging, data merging and time-series visualization plots used to execute the initial data QC, at the MSA Headquarters. The Data Distribution task began after the MSA field data were quality controlled and included the process of reconfiguring the field data files for the MSA end users. MSA–Phase I, Volume 1 described the Data Processing, in detail.[1] This report focuses on the subsequent Data Distribution Tool, developed in FY14.

The MSA–Phase I Data Distribution Tool consists of 4 functions: Data Storage, Data Extraction, Data Visualizations, and Data Documentation.

The Phase I-Data Storage was designed as a Windows MySQL database. The development of this Phase I-Data Storage function was broken into 3 steps. Due to delays in the software acquisition and installation processes, the initial data storage tool utilized the available Microsoft Access[4] database. A Visual Basic Application script was used in Access to read and manage all the MSA Merged Data files.

As the intermediate step, the "Proof of Concept" data storage was reinstalled in a UNIX-based MySQL environment. To load the data, a Visual Basic Application script was created to read all the MSA Merged Data files and write them into a formatted text file for each database table. The text files were transferred to the UNIX workstation, using the FTP. A MySQL LOAD command was then used to populate the tables. To improve efficiency, this process was automated.

The final step converted UNIX MySQL to Windows-based MySQL. At the time of this writing, the UNIX to Windows database conversion was completed; and, the tool to manage the data was in development.

Using the Phase I "Proof of Concept" data storage specifications, the current and future database size requirements were calculated. In 1 yr, for a single tower, the maximum amount of disk

storage needed for a merged data table would be approximately 128 MB. For 36 towers, the merged data table size would be approximately 5-GB per year. For 72 towers, approximately 10-GB per year would be required.

The subsequent 3 Data Distribution Tool functions were designed for execution by an MSA Operator, using desktop icons. To help visualize these 3 functions, an example of their usage from an operator's perspective follows:

> When a user submits a data request, the MSA Operator processes this request by clicking on the Data Management Extract function. The function initiates PuTTY software, a terminal emulator providing access to the UNIX environment required by MySQL database. The function then executes the extraction program *MSA Extract*. The Operator enters the requested specifications, which then extracts the MSA–Phase I data stored in a MySQL database. An option for choosing "all" variables is available, after which one could reduce the selection through a toggle interface. The data requested are instantaneously processed, and a sample of the results is shown on the screen. A Data Management FTP icon facilitates a UNIX-to-Windows data file transfer. The extracted numerical file is converted into color-coded time-series plots, using the Data Management GLE Plot icon. Clicking on the final icon, Data Management Excel Log and using local key strokes, the Operator tabulates the data request into an historical record of all MSA data requests received and processed, to date.

Lessons learned were gleaned throughout the MSA–Phase I Data Management "Proof of Concept" exercise. Four of these lessons included:

1) A significant amount of time was required to process Software Application Tool acquisition / installation requests.

2) Within the Data Management Tool, requesting multiple towers with 1 query for 1 output data file was very complex. This task was simplified by having separate queries and output files generated for each tower.

3) The Rapid Application Development Methodology was selected, based on the time constraints and the availability of fundamental algorithms needed to build the database queries. This application used a command line interface, which does not easily accommodate user entry errors. Using a GUI would make the interactions more user-friendly.

4) While the current Data Management Tool successfully proved the initial data management concept, utilizing a Web-based application would support GUI technology (reducing command line interface limitations) and add a greater potential for reaching more MSA users.

Looking forward, one of the next tasks to be coupled with this management tool is the attachment of detailed measurement site and sensor data format descriptions. These descriptive resources will be addressed in future reports.

As previously stated, data management is a nontrivial, critical pillar of a MSA. With software technologies advancing daily, the possibilities for improved user-friendly, efficient, and informative tools are almost open ended. With this report, a foundation for creating a successful MSA Data Management Tool has been initiated.

5. References and Notes

1. Vaucher GT, Swanson J, Raby J, Foley T. Harrison S, Brice R. D'Arcy S, Creegan E. Meteorological sensor array (MSA)–phase I, volume 1 ("proof of concept" overview). White Sands Missile Range (NM): Army Research Laboratory (US); September 2014. Report No.: ARL-TR-7058. Also available at http://www.arl.army.mil/www/default.cfm?technical_report=7179.

2. National Research Council (NRC) (US). When weather matters, science and service to meet critical societal needs. Washington DC: National Academies Press (US); 2010.

3 National Center for Atmospheric Research (NCAR) (US): Developmental Testbed Center (US). Model evaluation tools user's guide. Ver. 4.1 (METv4.1). Boulder (CO); 2013.

4. Microsoft Access is a registered trademark of Microsoft Corporation in the United States and/or other countries.

Appendix A. Program to Load Meteorological Sensor Array (MSA) Tables

This appendix appears in its original form, without editorial change.

Description: Program *BtnLoadMSATables* reads the Merged Data into the Microsoft Access tables.

```
Private Sub BtnLoadMSATables_Click()

'SUBROUTINE NAME:  BtnLoadMSATables
'                 AUTHOR:  Sandra Harrison
'       LAST REVISION:  July 15, 2014
'           DESCRIPTION:  This routine reads MSA merged data files from Drive L and stores
'                         the data from 3-17-14 to 5-12-14 in Access database tables
'                         MSA_DATA_2014 and MSA_DATA_HEADER.

   Dim Path, FileName, FileNameTbl, Folder, strSQL As String
   Dim FSO, TextFile

   Const ForReading = 1, ForWriting = 2, ForAppending = 8

   rootPath = "L:\MSA_PoC_Exercise\"
   dataPath = "MSA_Data\"

   FileID = 0

   yearString = "14"
   monString = "03"
   dayString = "17"
   numTowers = 5
   TowerString = Array("0101", "0102", "0202", "0302", "0103")
   utcString = "00"
   yearStringFirst = yearString
   monStringFirst = monString
   dayStringFirst = dayString

   Dim dbs As DAO.Database
   Dim rs As DAO.Recordset
   Dim rs1 As DAO.Recordset

   Set dbs = CurrentDb

   SQLstr = "SELECT * FROM MSA_Data;"
   Set rs = dbs.OpenRecordset(SQLstr)

   SQLstr = "SELECT * FROM MSA_Data_Header;"
   Set rs1 = dbs.OpenRecordset(SQLstr)

   Set FSO = CreateObject("Scripting.FileSystemObject")

   If FSO.FileExists(Path) Then
      Set TextFile = FSO.OpenTextFile(Path, 1, True)       '***Open hourly file for reading***
```

```
End If

currentLineNumber = 0
UTCCol = 1
MSTCol = 2
PressureCol = 3
RHCol = 4
InsolCol = 5
Temp2mCol = 6
WS2mCol = 7
WD2mCol = 8
U2mCol = 9
V2mCol = 10
W2mCol = 11
TS2mCol = 12
SoS2mCol = 13
Err2mCol = 14
Temp10mCol = 15
WS10mCol = 16
WD10mCol = 17
U10mCol = 18
V10mCol = 19
W10mCol = 20
TS10mCol = 21
SoS10mCol = 22
Err10mCol = 23
BVoltCol = 24
PTempCol = 25

YMD = yearString & monString & dayString
YMDFirst = YMD

TextDataFileOpen = 0
TextHeaderFileOpen = 0

For k = 0 To 56  '*** data from 3-17 to 5-12 in MSA PoC Exercise dir ***

  For i = 0 To numTowers - 1

    utcString = "00"
    dayString = dayStringFirst
    monString = monStringFirst
    yearString = yearStringFirst
    YMD = YMDFirst
    TowerDataFiles = False
```

```
    For j = 0 To 23

        '*** Open hourly sensor merged data files ***
        Continue = True
        sourceFileString = rootPath & dataPath & YMD & "merged\20" & yearString &
monString & dayString & "_" & utcString & "00" & "_" & TowerString(i) & "_merged.txt"
        FileName = "20" & yearString & monString & dayString & "_" & utcString & "00" &
"_" & TowerString(i) & "_merged.txt"
        If FSO.FileExists(sourceFileString) Then
           Set sourceFile = FSO.OpenTextFile(sourceFileString, ForReading, True)
           'MsgBox("Data File:  " & sourceFileString)
           TowerDataFiles = True
        Else
           'MsgBox("WARNING: File Not Found" & sourceFileString)
           'WScript.Quit
           Continue = False
        End If

    '*** Merge all hourly files into one ***
       If Continue = True Then

          currentLineNumber = 0

          Do While Not (sourceFile.AtEndofStream)
             CurCol = 0
             LogicalCol = 0
             currentLineNumber = currentLineNumber + 1

             currentLine = sourceFile.ReadLine
             currentLineArray = Split(currentLine, " ")

             If currentLineNumber = 1 Then
                FileDate = currentLineArray(0)
                JulianDay = currentLineArray(1)
                StartTimeUTC = currentLineArray(2)
                ConversionMST = currentLineArray(3)
                TowerReference = currentLineArray(4)
                Latitude = currentLineArray(5)
                Longitude = currentLineArray(6)
                Elevation = currentLineArray(7)

                rs1.AddNew            '***Add Header Record***
                rs1("File_Name") = FileName
                rs1("File_Date") = FileDate
                rs1("Julian_Day") = JulianDay
                rs1("Start_Time_UTC") = StartTimeUTC
```

```
            rs1("Conversion_To_MST") = ConversionMST
            rs1("Tower_Reference") = TowerReference
            rs1("Latitude") = Latitude
            rs1("Longitude") = Longitude
            rs1("Elevation") = Elevation
            rs1.Update
          Else
           For n = 0 To 24        '***Read each line of file accounting for extra spaces***
            Do While currentLineArray(CurCol) = "" Or currentLineArray(CurCol) =
Space(1)
              CurCol = CurCol + 1
            Loop

            LogicalCol = LogicalCol + 1

            If UTCCol = LogicalCol Then
              UTC_DecHrs = currentLineArray(CurCol)
              Min_Sec = UTC_DecHrs
              Hr = Left(StartTimeUTC, 2)
              Min_Sec = (CDec(Min_Sec) - CInt(Hr)) * 60
              If Min_Sec = 0 Then
                Min = "00"
                Sec = "00"
              Else
                Min = Int(Min_Sec)
                Sec = Min_Sec - Min
                Sec_Msec = Sec * 60
                Sec = Int(Sec_Msec)
                Msec = (Sec_Msec - Sec) * 10
                If Msec >= 5 Then
                  Sec = Sec + 1
                End If
                If Sec >= 5 Then
                  Min = Min + 1
                End If
                Sec = 0
              End If
              UTCyearString = Left(FileDate, 4)
              UTCmonString = Mid(FileDate, 5, 2)
              UTCdayString = Mid(FileDate, 7, 2)
              UTC_TimeStamp = UTCmonString & "/" & UTCdayString & "/" &
UTCyearString & " " & Hr & ":" & Min & ":" & Sec

            ElseIf MSTCol = LogicalCol Then
              MST_DecHrs = currentLineArray(CurCol)
              Min_Sec = MST_DecHrs
```

```
                Hr = Int(Min_Sec)
                Min_Sec = (CDec(Min_Sec) - Hr) * 60
                If Min_Sec = 0 Then
                   Min = "00"
                   Sec = "00"
                Else
                   Min = Int(Min_Sec)
                   Sec = Min_Sec - Min
                   Sec_Msec = Sec * 60
                   Sec = Int(Sec_Msec)
                   Msec = (Sec_Msec - Sec) * 10
                   If Msec >= 5 Then
                      Sec = Sec + 1
                   End If
                   If Sec >= 5 Then
                      Min = Min + 1
                   End If
                   Sec = 0
                End If

                MSTyearString = Left(FileDate, 4)
                MSTmonString = Mid(FileDate, 5, 2)
                MSTdayString = Mid(FileDate, 7, 2)

                If Hr > CInt(Left(StartTimeUTC, 2)) Then
                   Result = YMDFromJulian(MSTyearString, MSTmonString, MSTdayString,
CInt(JulianDay) - 1)
                End If

                MST_TimeStamp = MSTmonString & "/" & MSTdayString & "/" &
MSTyearString & " " & Hr & ":" & Min & ":" & Sec


             ElseIf PressureCol = LogicalCol Then
                Press2m = currentLineArray(CurCol)

             ElseIf RHCol = LogicalCol Then
                RelHum2m = currentLineArray(CurCol)

             ElseIf InsolCol = LogicalCol Then
                Insolation2m = currentLineArray(CurCol)

             ElseIf Temp2mCol = LogicalCol Then
                Temp2m = currentLineArray(CurCol)

             ElseIf WS2mCol = LogicalCol Then
```

```
    WindSpeed2m = currentLineArray(CurCol)

ElseIf WD2mCol = LogicalCol Then
    WindDir2m = currentLineArray(CurCol)

ElseIf U2mCol = LogicalCol Then
    UComp2m = currentLineArray(CurCol)

ElseIf V2mCol = LogicalCol Then
    VComp2m = currentLineArray(CurCol)

ElseIf W2mCol = LogicalCol Then
    WComp2m = currentLineArray(CurCol)

ElseIf TS2mCol = LogicalCol Then
    TempSonic2m = currentLineArray(CurCol)

ElseIf SoS2mCol = LogicalCol Then
    SoS2m = currentLineArray(CurCol)

ElseIf Err2mCol = LogicalCol Then
    Errors2m = currentLineArray(CurCol)

ElseIf Temp10mCol = LogicalCol Then
    Temp10m = currentLineArray(CurCol)

ElseIf WS10mCol = LogicalCol Then
    WindSpeed10m = currentLineArray(CurCol)

ElseIf WD10mCol = LogicalCol Then
    WindDir10m = currentLineArray(CurCol)

ElseIf U10mCol = LogicalCol Then
    UComp10m = currentLineArray(CurCol)

ElseIf V10mCol = LogicalCol Then
    VComp10m = currentLineArray(CurCol)

ElseIf W10mCol = LogicalCol Then
    WComp10m = currentLineArray(CurCol)

ElseIf TS10mCol = LogicalCol Then
    TempSonic10m = currentLineArray(CurCol)

ElseIf SoS10mCol = LogicalCol Then
    SoS10m = currentLineArray(CurCol)
```

```
  ElseIf Err10mCol = LogicalCol Then
     Errors10m = currentLineArray(CurCol)

  ElseIf BVoltCol = LogicalCol Then
     BatteryVolt = currentLineArray(CurCol)

  ElseIf PTempCol = LogicalCol Then
     PanelTemp = currentLineArray(CurCol)

  End If

  CurCol = CurCol + 1
Next  '*** For parsing 25 parameters per line ***

FileID = FileID + 1
rs.AddNew            '***Add Parameter Record***
rs("File_ID") = FileID
rs("File_Name") = FileName
rs("Tower_Reference") = TowerReference
rs("Universal_TimeStamp") = UTC_TimeStamp
rs("Universal_DecHrs") = UTC_DecHrs
rs("Mtn_Std_TimeStamp") = MST_TimeStamp
rs("Mtn_Std_DecHrs") = MST_DecHrs
rs("Pressure_2m") = Press2m
rs("Relative_Humidity_2m") = RelHum2m
rs("Insolation_2m") = Insolation2m
rs("Temperature_2m") = Temp2m
rs("Wind_Speed_2m") = WindSpeed2m
rs("Wind_Direction_2m") = WindDir2m
rs("U_Component_2m") = UComp2m
rs("V_Component_2m") = VComp2m
rs("W_Component_2m") = WComp2m
rs("Temperature_Sonic_2m") = TempSonic2m
rs("Speed_Of_Sound_2m") = SoS2m
rs("Errors_In_Avg_2m") = Errors2m
rs("Temperature_10m") = Temp10m
rs("Wind_Speed_10m") = WindSpeed10m
rs("Wind_Direction_10m") = WindDir10m
rs("U_Component_10m") = UComp10m
rs("V_Component_10m") = VComp10m
rs("W_Component_10m") = WComp10m
rs("Temperature_Sonic_10m") = TempSonic10m
rs("Speed_Of_Sound_10m") = SoS10m
rs("Errors_In_Avg_10m") = Errors10m
rs("Battery_Voltage") = BatteryVolt
```

```
                rs("Panel_Temp") = PanelTemp
                rs.Update

              End If
            Loop    '*** Do While Not AtEndOfStream ***

            sourceFile.Close

          End If  '*** Continue ***

'*** Set variables so we have the correct file name and directory of next hourly file to read ***
          utcString = CStr(CInt(utcString) + 1)
          utcString = String(2 - Len(utcString), "0") & utcString
          If utcString = "24" Then
            utcString = "00"

            If TowerDataFiles = True Then    '*** If no data files to read julian day from, then use
julian day from previous tower file which has already been incremented to next day ***
              JulianDay = JulianDay + 1
            End If

            Result = YMDFromJulian(yearString, monString, dayString, JulianDay)
            YMD = yearString & monString & dayString
            'MsgBox("Next day: " & yearString & monString & dayString)
          End If

        Next    '***For Loop Hour (j) ***

      Next  '***For Loop Tower (i) ***

      YMDFirst = YMD
      dayStringFirst = dayString
      monStringFirst = monString
      yearStringFirst = yearString

    Next    '*** For Loop YMD (k) ***

    rs.Close
    rs1.Close
    dbs.Close

End Sub
```

INTENTIONALLY LEFT BLANK.

Appendix B. Program to Load Text Files

This appendix appears in its original form, without editorial change.

Description: Program *BtnLoadTextFiles* reads the Merged Data into Text files to load the MySQL tables.

```
Private Sub BtnLoadTextFiles_Click()


'SUBROUTINE NAME:  BtnLoadTextFiles
'                 AUTHOR:  Sandra Harrison
'       LAST REVISION:  July 23, 2014
'           DESCRIPTION:  This routine reads MSA merged data files from the L Drive and
'                                 stores all the data from 3-17-14 to 5-12-14 in two text files.
'                                 One text file contains the header info and the other the data.

    Dim Path, FileName, strSQL As String

    Const ForReading = 1, ForWriting = 2, ForAppending = 8

    rootPath = "L:\MSA_PoC_Exercise\"
    dataPath = "MSA_Data\"

    FileID = 0
    yearString = "14"
    monString = "03"
    dayString = "17"
    numTowers = 5
    TowerString = Array("0101", "0102", "0202", "0302", "0103")
    utcString = "00"
    yearStringFirst = yearString
    monStringFirst = monString
    dayStringFirst = dayString


    currentLineNumber = 0
    UTCCol = 1
    MSTCol = 2
    PressureCol = 3
    RHCol = 4
    InsolCol = 5
    Temp2mCol = 6
    WS2mCol = 7
    WD2mCol = 8
    U2mCol = 9
    V2mCol = 10
    W2mCol = 11
    TS2mCol = 12
    SoS2mCol = 13
```

```
   Err2mCol = 14
   Temp10mCol = 15
   WS10mCol = 16
   WD10mCol = 17
   U10mCol = 18
   V10mCol = 19
   W10mCol = 20
   TS10mCol = 21
   SoS10mCol = 22
   Err10mCol = 23
   BVoltCol = 24
   PTempCol = 25

   YMD = yearString & monString & dayString
   YMDFirst = YMD

   TextDataFileOpen = 0
   TextHeaderFileOpen = 0

   Set FSO = CreateObject("Scripting.FileSystemObject")

   For k = 0 To 56  '*** data from 3-17 to 5-12 in MSA PoC Exercise dir ***

     For i = 0 To numTowers - 1

       utcString = "00"
       dayString = dayStringFirst
       monString = monStringFirst
       yearString = yearStringFirst
       YMD = YMDFirst
       TowerDataFiles = False

       For j = 0 To 23

          '*** Open hourly sensor merged data files ***
          Continue = True
          sourceFileString = rootPath & dataPath & YMD & "merged\20" & yearString &
monString & dayString & "_" & utcString & "00" & "_" & TowerString(i) & "_merged.txt"
          FileName = "20" & yearString & monString & dayString & "_" & utcString & "00" &
"_" & TowerString(i) & "_merged.txt"
          TextDataFileString = "C:\LRx_W\DB\Text Data Files\MSA_Data_140317_140512.txt"
          TextHeaderFileString = "C:\LRx_W\DB\Text Data
Files\MSA_Data_Header_140317_140512.txt"
          If FSO.FileExists(sourceFileString) Then
             Set sourceFile = FSO.OpenTextFile(sourceFileString, ForReading, True)
             'MsgBox("Data File:  " & sourceFileString)
```

```
            TowerDataFiles = True
        Else
            'MsgBox("WARNING: File Not Found" & sourceFileString)
            'WScript.Quit
            Continue = False
        End If


    '*** Merge all hourly files into one ***
        If Continue = True Then
            If TextDataFileOpen = 0 Then
                Set TextDataFile = FSO.OpenTextFile(TextDataFileString, ForWriting, True)
                TextDataFileOpen = 1
            Else
                TextDataFile.Close
                Set TextDataFile = FSO.OpenTextFile(TextDataFileString, ForAppending, True)
                TextDataFileOpen = 1
            End If

            If TextHeaderFileOpen = 0 Then
                Set TextHeaderFile = FSO.OpenTextFile(TextHeaderFileString, ForWriting, True)
                TextHeaderFileOpen = 1
            Else
                TextHeaderFile.Close
                Set TextHeaderFile = FSO.OpenTextFile(TextHeaderFileString, ForAppending,
True)
                TextHeaderFileOpen = 1
            End If

            currentLineNumber = 0

            Do While Not (sourceFile.AtEndofStream)
                CurCol = 0
                LogicalCol = 0
                currentLineNumber = currentLineNumber + 1

                currentLine = sourceFile.ReadLine
                currentLineArray = Split(currentLine, " ")

                If currentLineNumber = 1 Then       '***Write CSV header text file***
                    FileDate = currentLineArray(0)
                    JulianDay = currentLineArray(1)
                    StartTimeUTC = currentLineArray(2)
                    ConversionMST = currentLineArray(3)
                    TowerReference = currentLineArray(4)
                    Latitude = currentLineArray(5)
```

```
            Longitude = currentLineArray(6)
            Elevation = currentLineArray(7)

            TextHeaderFile.Write (FileName & ",")
            For a = 0 To 7
               If a = 7 Then
                  TextHeaderFile.WriteLine (currentLineArray(a))
               Else
                  TextHeaderFile.Write (currentLineArray(a) & ",")
               End If
            Next

         Else
          For n = 0 To 24        '***Read each line of file accounting for extra spaces***
            Do While currentLineArray(CurCol) = "" Or currentLineArray(CurCol) =
Space(1)
               CurCol = CurCol + 1
            Loop

            LogicalCol = LogicalCol + 1

            If UTCCol = LogicalCol Then
               UTC_DecHrs = currentLineArray(CurCol)
               Min_Sec = UTC_DecHrs
               Hr = Left(StartTimeUTC, 2)
               Min_Sec = (CDec(Min_Sec) - CInt(Hr)) * 60
               If Min_Sec = 0 Then
                  Min = "00"
                  Sec = "00"
               Else
                  Min = Int(Min_Sec)
                  Sec = Min_Sec - Min
                  Sec_Msec = Sec * 60
                  Sec = Int(Sec_Msec)
                  Msec = (Sec_Msec - Sec) * 10
                  If Msec >= 5 Then
                     Sec = Sec + 1
                  End If
                  If Sec >= 5 Then
                     Min = Min + 1
                  End If
                  Sec = 0
               End If
               UTCyearString = Left(FileDate, 4)              '***Write CSV data text file***
               UTCmonString = Mid(FileDate, 5, 2)
               UTCdayString = Mid(FileDate, 7, 2)
```

```
                UTC_Time = UTCyearString & "-" & UTCmonString & "-" & UTCdayString
& " " & Hr & ":" & Min & ":" & Sec

            ElseIf MSTCol = LogicalCol Then
                MST_DecHrs = currentLineArray(CurCol)
                Min_Sec = MST_DecHrs
                Hr = Int(Min_Sec)
                Min_Sec = (CDec(Min_Sec) - Hr) * 60
                If Min_Sec = 0 Then
                    Min = "00"
                    Sec = "00"
                Else
                    Min = Int(Min_Sec)
                    Sec = Min_Sec - Min
                    Sec_Msec = Sec * 60
                    Sec = Int(Sec_Msec)
                    Msec = (Sec_Msec - Sec) * 10
                    If Msec >= 5 Then
                        Sec = Sec + 1
                    End If
                    If Sec >= 5 Then
                        Min = Min + 1
                    End If
                    Sec = 0
                End If

                MSTyearString = Left(FileDate, 4)
                MSTmonString = Mid(FileDate, 5, 2)
                MSTdayString = Mid(FileDate, 7, 2)

                If Hr > CInt(Left(StartTimeUTC, 2)) Then
                    Result = YMDFromJulian(MSTyearString, MSTmonString, MSTdayString,
CInt(JulianDay) - 1)
                End If

                Mtn_Std_Time = MSTyearString & "-" & MSTmonString & "-" &
MSTdayString & " " & Hr & ":" & Min & ":" & Sec

            ElseIf PressureCol = LogicalCol Then
                Press2m = currentLineArray(CurCol)

            ElseIf Temp2mCol = LogicalCol Then
                Temp2m = currentLineArray(CurCol)

            ElseIf RHCol = LogicalCol Then
                RelHum2m = currentLineArray(CurCol)
```

```
ElseIf InsolCol = LogicalCol Then
   Insolation2m = currentLineArray(CurCol)

ElseIf WS2mCol = LogicalCol Then
   WindSpeed2m = currentLineArray(CurCol)

ElseIf WD2mCol = LogicalCol Then
   WindDir2m = currentLineArray(CurCol)

ElseIf U2mCol = LogicalCol Then
   UComp2m = currentLineArray(CurCol)

ElseIf V2mCol = LogicalCol Then
   VComp2m = currentLineArray(CurCol)

ElseIf W2mCol = LogicalCol Then
   WComp2m = currentLineArray(CurCol)

ElseIf Err2mCol = LogicalCol Then
   Errors2m = currentLineArray(CurCol)

ElseIf TS2mCol = LogicalCol Then
   TempSonic2m = currentLineArray(CurCol)

ElseIf SoS2mCol = LogicalCol Then
   SoS2m = currentLineArray(CurCol)

ElseIf Temp10mCol = LogicalCol Then
   Temp10m = currentLineArray(CurCol)

ElseIf WS10mCol = LogicalCol Then
   WindSpeed10m = currentLineArray(CurCol)

ElseIf WD10mCol = LogicalCol Then
   WindDir10m = currentLineArray(CurCol)

ElseIf U10mCol = LogicalCol Then
   UComp10m = currentLineArray(CurCol)

ElseIf V10mCol = LogicalCol Then
   VComp10m = currentLineArray(CurCol)

ElseIf W10mCol = LogicalCol Then
   WComp10m = currentLineArray(CurCol)
```

```
                    ElseIf Err10mCol = LogicalCol Then
                       Errors10m = currentLineArray(CurCol)

                    ElseIf TS10mCol = LogicalCol Then
                       TempSonic10m = currentLineArray(CurCol)

                    ElseIf SoS10mCol = LogicalCol Then
                       SoS10m = currentLineArray(CurCol)

                    ElseIf BVoltCol = LogicalCol Then
                       BatteryVolt = currentLineArray(CurCol)

                    ElseIf PTempCol = LogicalCol Then
                       PanelTemp = currentLineArray(CurCol)

                    End If

                    CurCol = CurCol + 1
                  Next  '*** For parsing 25 parameters per line ***

                  FileID = FileID + 1
                  TextDataFile.Write (FileID & "," & FileName & "," & TowerReference & "," &
UTC_Time & "," & UTC_DecHrs & "," & Mtn_Std_Time & "," & MST_DecHrs & ",")
                  TextDataFile.Write (Press2m & "," & Temp2m & "," & RelHum2m & "," &
Insolation2m & "," & WindSpeed2m & "," & WindDir2m & "," & UComp2m & "," &
VComp2m & "," & WComp2m & "," & Errors2m & "," & TempSonic2m & "," & SoS2m & ",")
                  TextDataFile.Write (Temp10m & "," & WindSpeed10m & "," & WindDir10m &
"," & UComp10m & "," & VComp10m & "," & WComp10m & "," & Errors10m & "," &
TempSonic10m & "," & SoS10m & "," & BatteryVolt & ",")
                  TextDataFile.WriteLine (PanelTemp)

                End If
              Loop    '*** Do While Not AtEndOfStream ***

              sourceFile.Close

            End If  '*** Continue ***

'*** Set variables so we have the correct file name and directory of next hourly file to read ***
            utcString = CStr(CInt(utcString) + 1)
            utcString = String(2 - Len(utcString), "0") & utcString
            If utcString = "24" Then
               utcString = "00"

               If TowerDataFiles = True Then    '*** If no data files to read julian day from, then use
julian day from previous tower file which has already been incremented to next day ***
```

```
                JulianDay = JulianDay + 1
            End If

            Result = YMDFromJulian(yearString, monString, dayString, JulianDay)
            YMD = yearString & monString & dayString
            'MsgBox("Next day: " & yearString & monString & dayString)
          End If

        Next    '***For Loop Hour (j) ***

      Next  '***For Loop Tower (i) ***

      YMDFirst = YMD
      dayStringFirst = dayString
      monStringFirst = monString
      yearStringFirst = yearString

    Next    '*** For Loop YMD (k) ***

    TextHeaderFile.Close
    TextDataFile.Close

End Sub
```

INTENTIONALLY LEFT BLANK.

Appendix C. MSA_Plots_MST.vbs

This appendix appears in its original form, without editorial change.

Description: Program *MSA_Plots_MST.vbs* prompts the user for the data filename, then extracts the parameter information from the descriptive data filename and writes it to the first line of the text data file for the *MSA_Plots_MST.GLE* program to parse.

```
'SUBROUTINE NAME:  MSA_Plots_MST.vbs
'AUTHOR:           Sandra Harrison
'LAST REVISION:    Aug 6, 2014
'DESCRIPTION:      This routine prompts the user for the data filename. The Script then
'                  extracts the parameter information from the descriptive data filename,
'                  and writes it to the first line of the text data file, for
'                   MSA_Plots_MST.GLE to parse.

   Set wShell=CreateObject("WScript.Shell")
   Set oExec=wShell.Exec("mshta.exe ""about:<input type=file
id=FILE><script>FILE.click();new
ActiveXObject('Scripting.FileSystemObject').GetStandardStream(1).WriteLine(FILE.value);clos
e();resizeTo(0,0);</script>""")
   Path = oExec.StdOut.ReadLine
   'wscript.echo Path

   set FSO = CreateObject("Scripting.FileSystemObject")
   Set WshShell = CreateObject("WScript.Shell")

   rootPath = "C:\LRx_W\DB"
   applicationsPath = "Text_Data_Files"

   Const ForReading = 1, ForWriting = 2, ForAppending = 8

   FileName = StrReverse(Left(StrReverse(Path), InStr(1, StrReverse(Path), "\") - 1))
   Folder = Left(Path, InStrRev(Path, "\"))


   Start_Param_Index = InStr(1, FileName, "_")
   For i = 0 To 2
      Start_Param_Index = InStr(Start_Param_Index + 1, FileName, "_")
   Next


   If Start_Param_Index+1 <> 28 Then
      MsgBox ("Error in File Name Format Date/Time:
M_SYMMDDhhmm_EYMMDDhhmm_##_")
   End If

   Params_ext = Right(FileName, Len(FileName) - Start_Param_Index)
   Params = Left(Params_ext, InStrRev(Params_ext, ".") - 1)
```

```
   '***Use x for TempSonic instead of t for GLE since GLE gets confused with T and t in
string***
   Params = Replace (Params, "t", "x", 1, 2)

   Start2m = InStr(Params, "2")
   If Start2m <> 0 Then
      Params2m = Right(Params, Len(Params) - 1)
      Start10m = InStr(Params2m, "1")
      If Start10m <> 0 Then
         Params10m = Right(Params2m, Len(Params2m) - Start10m - 1)
         Params2m = Left(Params2m, Start10m - 2)
         StartMisc = InStrRev(Params10m, "_")
         If StartMisc <> 0 Then
            ParamsMisc = Right(Params10m, Len(Params10m) - StartMisc)
            Params10m = Left(Params10m, StartMisc - 1)
         Else
            ParamsMisc = "Misc"
         End If
      Else
         Params10m = "10"
         StartMisc = InStrRev(Params2m, "_")
         If StartMisc <> 0 Then
            ParamsMisc = Right(Params2m, Len(Params2m) - StartMisc)
            Params2m = Left(Params2m, StartMisc - 1)
         Else
            ParamsMisc = "Misc"
         End If
      End If
   Else
      Params2m = "2"
      Start10m = InStr(Params, "1")
      If Start10m <> 0 Then
         Params10m = Right(Params, Len(Params) - Start10m - 1)
         StartMisc = InStrRev(Params10m, "_")
         If StartMisc <> 0 Then
            ParamsMisc = Right(Params10m, Len(Params10m) - StartMisc)
            Params10m = Left(Params10m, StartMisc - 1)
         Else
            ParamsMisc = "Misc"
         End If
      Else
         Params10m = "10"
         ParamsMisc = Params
      End If
   End If
```

```
set sourceFile = FSO.OpenTextFile(Path, ForReading, True)
dataFileOpen = 0

Do While Not(sourceFile.AtEndofStream)
	If dataFileOpen = 0 Then
		set dataFile = FSO.OpenTextFile("MSA_Plots_MST.dat", ForWriting, True)
		dataFile.WriteLine("!" & Params2m & "_" & Params10m & "," & ParamsMisc)
		dataFile.close
		set dataFile = FSO.OpenTextFile("MSA_Plots_MST.dat", ForAppending, True)
		dataFileOpen = 1
	Else
		dataFile.close
		set dataFile = FSO.OpenTextFile("MSA_Plots_MST.dat", ForAppending, True)
		dataFileOpen = 1
	End If
	currentLine = sourceFile.ReadLine
	dataFile.WriteLine(currentLine)
Loop
sourceFile.close
dataFile.close


GLE_MSTScript = "MSA_Plots_MST.gle "
GLE_ProgramPath = "C:\Program Files (x86)\Gle4\bin\qgle "
'MsgBox(Params2m & "_" & Params10m & "," & ParamsMisc)

GLEScriptRunStringMST = GLE_ProgramPath & GLE_MSTScript
Set GLE_Exec_MST = WshShell.Exec(GLEScriptRunStringMST)
```

Appendix D. MSA_Plots_MST.GLE

This appendix appears in its original form, without editorial change.

Description: Program *MSA_Plots_MST.GLE* is called from *MSA_Plots_MST.vbs*. This program creates a 24- h midnight-to-midnight Mountain Standard Time (MST) data display of various Meteorological Sensor Array (MSA) data plots.

```
! PROGRAM NAME:        MSA_Plots_MST.GLE
! AUTHOR:              Sandra Harrison
! LAST REV:            09-09-2014, sh
! REQUIRED PROGRAM: This program is called from MSA_Plots_MST.vbs
! PURPOSE:             This program creates a 24 hour midnight to midnight MST data
!                      display of various plots for MSA Data.  A string in the first line of
!                      the data file is decoded to know what data and plots to display.

size 63 60
set font ssb
set hei 0.7
set alabelscale 1.0
set atitlescale 1.0
set titlescale 1.0

!Find YYYYMMDD and XX:YY from file and then set up
! file date string and tower position for title info

dataFile$ = "MSA_Plots_MST.dat"

fopen dataFile$ f1 read

!*** get first line of parameter values ***
fgetline f1 line$

!*** parse the line into 2m, 10m, and Misc params ***
Params$ = seg$(line$, 2, len(line$))
for i = 1 to len(Params$)
        Value$ = seg$(Params$, i, i)
        if Value$ = "_" then
                Params2m$ = seg$(Params$, 1, i-1)
                Start10m = i + 1
        else if Value$ = "," then
                Params10m$ = seg$(Params$, Start10m, i-1)
                ParamsMisc$ = seg$(Params$, i+1, len(Params$))
        end if
next i

fgetline f1 line$
fgetline f1 line$
fclose f1
```

```
year$ = seg$(line$, 1, 4)
mon$ = seg$(line$, 5, 6)
day$ = seg$(line$, 7, 8)
position$ = seg$(line$, 19, 23)

monValue = val(mon$)
xcomp$ = seg$(position$, 2, 2)
ycomp$ = seg$(position$, 5, 5)
xcompValue = val(xcomp$)
ycompValue = val(ycomp$)

if (xcompValue = 1) then
        if (ycompValue = 1) then
                tower$ = "1 "
        else if (ycompValue = 2) then
                tower$ = "2 "
        else
                tower$ = "5 "
        end if
else if (xcompValue = 2) then
        tower$ = "3 "
else
        tower$ = "4 "
end if

if (monValue = 1) then
   mon$ = "Jan "
else if (monValue = 2) then
   mon$ = "Feb "
else if (monValue = 3) then
   mon$ = "Mar "
else if (monValue = 4) then
   mon$ = "Apr "
else if (monValue = 5) then
   mon$ = "May "
else if (monValue = 6) then
   mon$ = "Jun "
else if (monValue = 7) then
   mon$ = "Jul "
else if (monValue = 8) then
   mon$ = "Aug "
else if (monValue = 9) then
   mon$ = "Sep "
else if (monValue = 10) then
   mon$ = "Oct "
```

```
else if (monValue = 11) then
   mon$ = "Nov "
else if (monValue = 12) then
   mon$ = "Dec "
end if


!Output file date and tower position header
set hei 1.0
set color blue
amove 8 57.5
write "Data Date: " mon$ day$ ", " year$ " MST"
amove 42 57.5
write "Tower " tower$  "(" position$ ")"
set color black
set hei 0.7


if Params2m$ <> "2" then
  for i = 1 to len(Params2m$)
     Value$ = seg$(Params2m$,i,i)

     if Value$ = "P" then
        Press2m = 1
        P2Col = i + 4

     else if Value$ = "T" then
        Temp2m = 1
        T2Col = i + 4

     else if Value$ = "R" then
        RH = 1
        R2Col = i + 4

     else if Value$ = "I" then
        Insol = 1
        I2Col = i + 4

     else if Value$ = "S" then
        WS2m = 1
        S2Col = i + 4

     else if Value$ = "D" then
        WD2m = 1
        D2Col = i + 4
```

```
      else if Value$ = "u" then
        U2m = 1
        u2Col = i + 4

      else if Value$ = "v" then
        V2m = 1
        v2Col = i + 4

      else if Value$ = "w" then
        W2m = 1
        w2Col = i + 4

      else if Value$ = "e" then
        Errs2m = 1
        e2Col = i + 4

      else if Value$ = "x" then
        tSonic2m = i
        t2Col = i + 4

      else if Value$ = "c" then
        CC2m = 1
        c2Col = i + 4

      end if

    next i
  end if


  if Params10m$ <> "10" then
    for i = 1 to len(Params10m$)
      Value$ = seg$(Params10m$,i,i)

      if Value$ = "T" then
        Temp10m = 1
        T10Col = i + 4 + len(Params2m$)

      else if Value$ = "S" then
        WS10m = 1
        S10Col = i + 4 + len(Params2m$)

      else if Value$ = "D" then
        WD10m = 1
        D10Col = i + 4 + len(Params2m$)
```

```
      else if Value$ = "u" then
        U10m = 1
        u10Col = i + 4 + len(Params2m$)

      else if Value$ = "v" then
        V10m = 1
        v10Col = i + 4 + len(Params2m$)

      else if Value$ = "w" then
        W10m = 1
        w10Col = i + 4 + len(Params2m$)

      else if Value$ = "e" then
        Errs10m = 1
        e10Col = i + 4 + len(Params2m$)

      else if Value$ = "x" then
        tSonic10m = 1
        t10Col = i + 4 + len(Params2m$)

      else if Value$ = "c" then
        CC10m = 1
        c10Col = i + 4 + len(Params2m$)
      end if

    next i
  end if


  if ParamsMisc$ <> "Misc" then
    for i = 1 to len(ParamsMisc$)
      Value$ = seg$(ParamsMisc$,i,i)

      if Value$ = "B" then
        BV = 1
        BVCol = i + 4 + len(Params2m$) + len(Params10m$)

      else if Value$ = "P" then
        PT = 1
        PTCol = i + 4 + len(Params2m$) + len(Params10m$)
      end if

    next i
  end if
```

```
!Relative humidity and station Air Pressure plot at top of page

if Press2m = 1 and RH = 1 then

amove 43 44
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 100
YAxisMin = 0
YAxisMajorTick = 10
YAxisSubTick = 5.0

begin graph
  size 20 12
  nobox
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .5
  y2ticks length .5
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  ytitle "Relative Humidity (percent)"
  xtitle "MST (decimal hours)" dist 0.5
  y2axis min 850 max 880 dticks 10 dsubticks 0
  y2title "Station Air Pressure (mb)"
  y2labels on
  data dataFile$ d1=c3,c[eval("R2Col")] d2=c3,c[eval("P2Col")]
  let d2 = (d2-850)*3.3333
  d1 lstyle 1 lwidth .04 color brown
  d2 lstyle 1 lwidth .04 color blue
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "Humidity" lstyle 1 lwidth .04 color brown
  separator
  text "Pressure" lstyle 1 lwidth .04 color blue
end key
set hei 0.7

!Output graph title for pressure and relative humidity
amove 45 54.8
```

```
set color green
write "RH (%) \; Pressure (mb) \; Tower " tower$ "(" position$ ")"
set color black

end if


!Wind speed plot at top of page

if WS2m = 1 and WS10m = 1 then

amove 1 30
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6
XAxisSubTick = 1
YAxisMax = 20
YAxisMax30 = 30
YAxisMin = 0
YAxisMajorTick = 5
YAxisSubTick = 1

begin graph
  size 20 12
  nobox
  !title "Wind Speed (1 - Minute Average)"
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length 0.3
  y2ticks length 0.3
  y2subticks length 0.3
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  y2axis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick on
  ylabels on
  y2labels off
  ytitle "Wind Speed (m/s)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c3,c[eval("S2Col")] d2=c3,c[eval("S10Col")]
  d1 lstyle 1 color brown
  d2 lstyle 1 color blue
end graph

begin key
```

```
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2m" lstyle 1 lwidth .04 color brown
  separator
  text "10m" lstyle 1 lwidth .04 color blue
end key
set hei 0.7

!Output Graph Title wind speed
amove 5 40.8
set color green
write "Wind Speed (m/s) \; Tower " tower$ "(" position$ ")"
set color black

end if


!Wind direction plot at top of page

if WD2m = 1 and WD10m = 1 then

amove 1 44
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6
XAxisSubTick = 1
YAxisMax = 360
YAxisMin = 0
YAxisMajorTick = 90
YAxisSubTick = 30

begin graph
  size 20 12
  nobox
  !title "Wind Direction (1 - Minute Average)"
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .2
  yticks length .2
  ysubticks length 0.3
  ylabels on
  y2labels off
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick grid dsubticks YAxisSubTick
  ytitle "Wind Direction (degrees)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c3,c[eval("D2Col")] d2=c3,c[eval("D10Col")]
```

```
  d1 marker circle msize 0.3 color brown
  d2 marker circle msize 0.3 color blue
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2m" lstyle 1 lwidth .04 color brown
  separator
  text "10m" lstyle 1 lwidth .04 color blue
end key
set hei 0.7

!Output Graph Title wind direction
amove 3.5 54.8
set color green
write "Wind Direction: (degrees) \; Tower " tower$ "(" position$ ")"
set color black

end if


!U plot at bottom of page

if U2m = 1 and U10m = 1 then

amove 1 16
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6
XAxisSubTick = 1
YAxisMax = 20
YAxisMax30 = 30
YAxisMin = -5
YAxisMajorTick = 5
YAxisSubTick = 1

begin graph
  size 20 12
  nobox
  !title "2m U Plot"
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
```

```
  ysubticks length 0.3
  y2ticks length 0.3
  y2subticks length 0.3
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  y2axis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick on
  ylabels on
  y2labels off
  ytitle "U (m/s)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c3,c[eval("u2Col")] d2=c3,c[eval("u10Col")]
  let d3 = d1*0.0
  d1 lstyle 1 color brown
  d2 lstyle 1 color blue
  d3 lstyle 1 lwidth .1 color deeppink
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2m" lstyle 1 lwidth .04 color brown
  separator
  text "10m" lstyle 1 lwidth .04 color blue
end key
set hei 0.7

!Output Graph Title of U Component
amove 5 26.8
set color green
write "U Component (m/s) \; Tower " tower$ "(" position$ ")"
set color black

amove 20 26.8
write u2col u10Col

end if


!V plot at bottom of page

if V2m = 1 and V10m = 1 then

amove 22 16
XAxisMax = 24
XAxisMin = 0
```

```
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 10
YAxisMax30 = 30
YAxisMin = -10
YAxisMajorTick = 5
YAxisSubTick = 1

begin graph
  size 20 12
  nobox
  !title "2m V Plot"
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length 0.3
  y2ticks length 0.3
  y2subticks length 0.3
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  y2axis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick on
  ylabels on
  y2labels off
  ytitle "V (m/s)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c3,c[eval("v2Col")] d2=c3,c[eval("v10Col")]
  let d3 = d1*0.0
  d1 lstyle 1 color brown
  d2 lstyle 1 color blue
  d3 lstyle 1 lwidth .1 color deeppink
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2m" lstyle 1 lwidth .04 color brown
  separator
  text "10m" lstyle 1 lwidth .04 color blue
end key
set hei 0.7

!Output Graph Title for V Component
amove 25.5 26.8
set color green
```

```
write "V Component (m/s) \; Tower " tower$ "(" position$ ")"
set color black

end if


!W plot at bottom of page

if W2m = 1 and W10m = 1 then

amove 43 16
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 1
YAxisMax30 = 30
YAxisMin = -1
YAxisMajorTick = .5
YAxisSubTick = .1

begin graph
  size 20 12
  nobox
  !title "2m W Plot"
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length 0.3
  y2ticks length 0.3
  y2subticks length 0.3
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  y2axis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick on
  ylabels on
  y2labels off
  ytitle "W (m/s)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c3,c[eval("w2Col")] d2=c3,c[eval("w10Col")]
  let d3 = d1*0.0
  d1 lstyle 1 color brown
  d2 lstyle 1 color blue
  d3 lstyle 1 lwidth .1 color deeppink
end graph
```

```
begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2m" lstyle 1 lwidth .04 color brown
  separator
  text "10m" lstyle 1 lwidth .04 color blue
end key
set hei 0.7

!Output Graph Title for W Component
amove 46.5 26.8
set color green
write "W Component (m/s) \; Tower " tower$ "(" position$ ")"
set color black

end if


!Errors plot at bottom of page

if Errs2m = 1 and Errs10m = 1 then

amove 1 2
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 1200
YAxisMax30 = 30
YAxisMin = 0
YAxisMajorTick = 300
YAxisSubTick = 100

begin graph
  size 20 12
  nobox
  !title "Errors"
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length 0.3
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  ylabels on
```

```
  ytitle "# of Errors"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c3,c[eval("e2Col")] d2=c3,c[eval("e10Col")]
  d1 lstyle 1 lwidth .05 color brown
  d2 lstyle 1 lwidth .05 color blue
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2m" lstyle 1 lwidth .04 color brown
  separator
  text "10m" lstyle 1 lwidth .04 color blue
end key
set hei 0.7

!Output Graph Title
amove 4 12.8
set color green
write "Number of Sonic Errors \; Tower " tower$ "(" position$ ")"
set color black

end if


!Battery Voltage and Temp-Sonic10m plot at bottom of page

if BV = 1 and tSonic10m = 1 then

amove 43 2
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 15
YAxisMax30 = 30
YAxisMin = 10
YAxisMajorTick = 1
YAxisSubTick = 0

begin graph
  size 20 12
  nobox
  !title "Plot"
  yaxis grid
```

```
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  !ysubticks length 0.3
  y2ticks length 0.5
  y2subticks length 0.5
  !yplaces 10 11 12 13 14 15 16
  !ynames "10" "11" "12" "13" "14" "15" "16"
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  y2axis min 0 max 60 dticks 12 dsubticks 6
  ylabels on
  y2labels on
  ytitle "Battery Voltage (V)"
  xtitle "MST (decimal hours)" dist 0.5
  y2title "T-Sonic10m ( ^{o}C)"
  data dataFile$ d1=c3,c[eval("BVCol")] d2=c3,c[eval("t10Col")]
  let d2 = (d2/12)+10
  d1 lstyle 1 lwidth .05 color brown
  d2 lstyle 1 lwidth .05 color blue
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "Battery" lstyle 1 lwidth .04 color brown
  separator
  text "T-Sonic10m" lstyle 1 lwidth .04 color blue
end key
set hei 0.7

!Output Graph Title
amove 45 12.8
set color green
write "Battery(V) \; T-Sonic10m(^{o}C) \; Tower " tower$ "(" position$ ")"
set color black

end if


!Temperature gradient plot at top of page

if Temp2m = 1 and Temp10m = 1 then

amove 22 30
```

```
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 1.0
YAxisMin = -0.5
YAxisMajorTick = 0.1
YAxisSubTick = 0.0

begin graph
  size 20 12
  nobox
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length 0.2
  !yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  ylabels on
  y2labels off
  ytitle "Temperature Gradient ( ^{o}C/m)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c3,c[eval("T2Col")] d2=c3,c[eval("T10Col")]
  let d3 = (d2-d1)/8
  let d4 = d1*0.0
  d3 lstyle 1 color purple
  d4 lstyle 1 lwidth .1 color deeppink
end graph

!Output graph title for temperature gradient
amove 25 40.8
set color green
write "Temp Gradient (^{o}C/m) \; Tower " tower$ "(" position$ ")"
set color black

end if


!Temperature plot at top of page

if Temp2m = 1 and Temp10m = 1 then

amove 22 44
XAxisMax = 24
XAxisMin = 0
```

```
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 35
YAxisMin = 0
YAxisMajorTick = 5
YAxisSubTick = 0

begin graph
  size 20 12
  nobox
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length .2
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick grid dsubticks YAxisSubTick
  !yaxis dticks YAxisMajorTick grid dsubticks YAxisSubTick
  ylabels on
  y2labels off
  ytitle "Temperature ( ^{o}C)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c3,c[eval("T2Col")] d2=c3,c[eval("T10Col")]
  d1 lstyle 1 color brown
  d2 lstyle 1 color blue
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2 m" lstyle 1 lwidth .04 color brown
  separator
  text "10 m" lstyle 1 lwidth .04 color blue
  end key
set hei 0.7

!Output Graph Title for Temperature
amove 26 54.8
set color green
write "Temperature (^{o}C) \; Tower " tower$ "(" position$ ")"
set color black

end if


!Solar irradiance plot at top of page
```

```
if Insol = 1 then

amove 43 30
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 1400
YAxisMin = 0
YAxisMajorTick = 200
YAxisSubTick = 100

begin graph
  size 20 12
  nobox
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .2
  yticks length .2
  ysubticks length 0.3
  ylabels on
  y2labels off
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  ytitle "Solar Irradiance (W/m^2)"
  xtitle "MST (decimal hours)" dist 0.5
  !data dataFile$ d1=c3,c7
  data dataFile$ d1=c3,c[eval("I2Col")]
  d1 lstyle 1 lwidth .04 color brown
end graph

!Output graph title for solar irradiance
amove 46 40.8
set color green
write "Solar Irradiance (W/m^2) \; Tower " tower$ "(" position$ ")"
set color black

end if


!Dew Point plot at bottom of page

if RH = 1 and Temp2m = 1 then

amove 22 2
XAxisMax = 24
XAxisMin = 0
```

```
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 0
YAxisMin = -15
!YAxisMajorTick = 10
YAxisMajorTick = 5
YAxisSubTick = 1

begin graph
  size 20 12
  nobox
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length .2
  !yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick grid dsubticks YAxisSubTick
  yaxis dticks YAxisMajorTick grid dsubticks YAxisSubTick
  ylabels on
  y2labels off
  ytitle "Temperature ( ^{o}C)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c3,c[eval("T2Col")] d2=c3,c[eval("R2Col")]
  let d4 = (d2*0.06112*EXP((17.67*d1)/(d1+243.5)))
  let d5 = ((243.5*LOG(d4/6.112))/(17.67-LOG(d4/6.112)))
  let d6 = d1*0.0
  d5 lstyle 1 lwidth .04 color green
  d6 lstyle 1 lwidth .1 color deeppink
end graph

!Output Graph Title for DewPoint
amove 27 12.8
set color green
write "DewPoint (^{o}C) \; Tower " tower$ "(" position$ ")"
set color black

end if
```

Appendix E. README File

This appendix appears in its original form, without editorial change.

Description: README file explains the extracted Meteorological Sensor Array (MSA) data file's name.

```
================================================================
Filenames and the user logs build from the following Parameter
codes in the chart below, to represent selections as follows:

                                                  Collapsed form
                                                  --------------
Start/End Date and Time: 2014-04-13 00:00:00       1404130000
                         2014-04-13 23:59:59       1404132359

                                                   Tower: 13

              Parameter Number:   000000000111   111111122 22
                                  123456789012   345678901 23
      Selected Parameter Codes: _2PTRISDuvwetc_10TSDuvwetc_BP

Filename w/ Collapsed Codes:
M_1404130000_1404132359_13_2PTRISDuvwetc_10TSDuvwetc_BP[V##].dat
|rdm

Log List Representation of the same:
PP02,TT02,RH02,SR02,WS02,WD02,UU02,VV02,WW02,ER02,TS02,CC02,TT10
,WS10,WD02,UU10,VV10,WW10,ER10,TS10,CC10,BaVo,PaTe

=========================Parameter Chart=========================
01.          Pressure_2m P PP02 13.       Temperature_10m T TT10
02.       Temperature_2m T TT02 14.        Wind_Speed_10m S WS10
03. Relative_Humidity_2m R RH02 15.    Wind_Direction_10m D WD02
04.        Insolation_2m I SR02 16.       u_Component_10m u UU10
05.        Wind_Speed_2m S WS02 17.       v_Component_10m v VV10
06.    Wind_Direction_2m D WD02 18.       w_Component_10m w WW10
07.       u_Component_2m u UU02 19.     errors_in_avg_10m e ER10
08.       v_Component_2m v VV02 20. temperature_sonic_10m t TS10
09.       w_Component_2m w WW02 21.    speed_of_sound_10m c CC10
10.     errors_in_avg_2m e ER02 22.       Battery_Voltage B BaVo
11. temperature_sonic_2m t TS02 23.            Panel_Temp P PaTe
12.    speed_of_sound_2m c CC02
================================================================
```

Appendix F. HELP File

This appendix appears in its original form, without editorial change.

Description: The HELP file gives a detailed explanation of how to use the Data Management Tool application, and the different options available.

```
This program extracts weather data from the MSA database based
on the selections below, and saves it in the Reports directory
(currently in the same directory as this program) under the
requestor initials.

Three files are saved. Their names look something like the
following:
M_1404130000_1404132359_13__2PTRIWDuvwec_10TWDuvwec_BP.dat
M_1404130000_1404132359_13__2PTRIWDuvwec_10TWDuvwec_BP.rdm
M_1404130000_1404132359_13__2PTRIWDuvwec_10TWDuvwec_BP.sql

The *.dat file holds the extracted data
The *.rdm file holds the translation of the long descriptive
filename
The *.sql file holds the SQL query used to generate the "*.dat"
file

  Registration:
      MSA Tech: INITIALS
      Requested by: INITIALS
  Backspace to remove char to left.

  Date/Time selection:
  Start Date: MM/DD/YY Time: hh:mm:ss
    End Date: MM/DD/YY Time: hh:mm:ss

  Enter digits for dates and times
  Backspace or Left arrow to remove char to left.
  Moving backward from Time to Date is not
  possible (yet). Pressing "q" exit this program.


  ======MSA Towers======
  1. 01:01
  2. 01:02
  3. 02:02
  4. 03:02
  5. 01:03
  ======================
  Select:
  5_____________________

Tower Selection:
Delimit multiple tower selections with commas, spaces, decimals
```

```
or dashes.  A dash between tower numbers generates an inclusive
range of towers to query. "a" for all selections anywhere on the
line indicates all towers will be queried.

Editing: Backspace to remove char to the left (left arrow will
not work here).

Extraneous chars will be removed; repeats will be removes and
selected towers will be recorded from least to greatest

Examples: 3 1-3, 4.10  translates to 1,2,3,4

Parameter Selection:
=================================================================
 01. Pressure_2m.........<--        13. Temperature_10m
 02. Temperature_2m                 14. Wind_Speed_10m
 03. Relative_Humidity_2m.<--       15. Wind_Direction_10m
 04. Insolation_2m.......<--        16. u_Component_10m
 05. Wind_Speed_2m.......<--        17. v_Component_10m
 06. Wind_Direction_2m              18. w_Component_10m
 07. u_Component_2m                 19. errors_in_avg_10m
 08. v_Component_2m                 20. temperature_sonic_10m
 09. w_Component_2m                 21. speed_of_sound_10m
 10. errors_in_avg_2m               22. Battery_Voltage
 11. temperature_sonic_2m           23. Panel_Temp
 12. speed_of_sound_2m
=================================================================
Select (<Enter> when done): _05_

Entering two digits will select a parameter and selected
parameters will be indicated with a "...<--" tag. Selecting a
parameter unselects it. Pressing "q" here also exit this
program.

When a lot of parameters need selecting, entering aa or al, for
all, will select all parameter and entering two digit for each
undesired parameter will un-select them.

Arriving here in static mode (see <F>reeze selection below),
means that "8." was selected in "<F>reeze selection" and instead
of only showing the query as is default in static mode, the
database will be queried, and result files will be generated.

End Menu:
-----------------------------------------------------------------
<E>nd, <C>ontinue, <F>reeze selection/repeat, <L>ogs, <H>elp
-----------------------------------------------------------------
```

<E>nd: exits the program
<C>ontinues: program after return from <F>reeze, <L>ogs, or
<H>elp
<F>reeze selection:

STATE: S

    1. Show selections
    2. Make current selections default
    3. Continue using current selections (ON)
    4.   Registration changes ALLOWED   (OFF)
    5.   Date/Time changes ALLOWED   (OFF)
    6.   Towers changes ALLOWED   (OFF)
    7.   Parameters changes ALLOWED   (OFF)
    8.   Queries changes ALLOWED   (OFF)
         Select (<Enter> when done): ___

By selecting "3." the current selection becomes static (frozen) and repeat runs, via the <C>ontinues, reproduce the same result. Until "3." is turned ON,"4 through 8." will not be visible.

Selecting "4 through 8." allow selections to be turned ON for user input or OFF for static use of the former selections. Pressing the <Enter> key returns the user to the "End Menu", above, so the "<C>ontinue" option can be selected to do another run. "q" entered here will quit the program.

Suppose the user needs 10 runs for a requestor. It becomes tedious to repeatedly re-enter the MSA and user info. The "<F>reeze selection" option allows selection of "5., 6., 7. 8" here so only those selections need entering. Or perhaps, only different sets of towers are desired with the same Date/Time, Parameter and query selection. This can be accomplished by selecting "6." here to allow changes on the selection screen while Registration, Date/Time, Parameters and Queries remain "frozen".

Perhaps the user wishes to save the current selection until another time. "2." accomplishes this by rewriting the default variables into this program and restarting it.

This program reads the MSA database headers for SQL query generation.

Should those headers be changed in the database, they need to be re-saved into this program by:

1. Take this program out of static mode "3."
2. Run a query
3. Select <F>reeze selection again
4. "1." can be selected to see what defaults will be saved and further runs with differing "frozen" states can correct runtime selections as desired.
5. Selecting "2." to write the variables into the program and restarts it.

<L>ogs: Presents a formatted view of previous queries by the requestor
<H>elp: View this help screen

INTENTIONALLY LEFT BLANK.

List of Symbols, Abbreviations, and Acronyms

AGL	above ground level
ARL	US Army Research Laboratory
ASCII	American Standard Code for Information Interchange
DAS	data acquisition system
FTP	File Transfer Protocol
FY	fiscal year
GLE	Graphics Layout Engine
GUI	graphical user interface
MB	megabyte
MET	Model Evaluation Tool
MSA	Meteorological Sensor Array
MST	Mountain Standard Time
MySQL	My Structured Query Language (MySQL = My[name of a developer's daughter] SQL)
NRC	National Research Council
NTP	Network Time Protocol
PV	Photovoltaic
QC	quality control
SSI	STS Systems Integration
STS	SpecPro Technical Services
V&V	validation and verification
WRE-N	Weather Running Estimate-Nowcast

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA

2 (PDF) DIRECTOR US ARMY RSRCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT

1 (PDF) GOVT PRINTG OFC A MALHOTRA

2 (CD) US ARMY RSRCH LAB ATTN RDRL CIE M BLDG 1622 WSMR NM 88002 D KNAPP J SMITH

4 (CD) US ARMY RSRCH LAB ATTN RDRL CIE D BLDG 1622 WSMR NM 88002 S HARRISON R HORNBAKER S OBRIEN R RANDALL

10 (5 CD, 5 HC) US ARMY RSRCH LAB G VAUCHER ATTN RDRL CIE D BLDG 1622 WSMR NM 88002

1 (CD) US ARMY RSRCH LAB P CLARK ATTN RDRL CIE 2800 POWDER MILL RD ADELPHI MD 20783-1138

1 (CD) DR J MCLAY NAVAL RESEARCH LABORATORY 7 GRACE HOPPER AVE STOP 2 MONTEREY CA 93943

1 (CD) R CRAIG DAF CIVILIAN HQ AFWA 2WXG 16WS/WXN 101 NELSON DRIVE OFFUTT AFB NE 68113-1023

1 (CD) ARMY JOINT SUPPORT TEAM SFAE IEW&S DCGS A ATTN G BARNES 238 HARSTON ST BLDG 90060 HURLBURT FIELD FL 32544

1 (CD) J STALEY ARMY WEATHER PROPONENT OFFICE INTEGRATION SYNCHRONIZATION AND ANALYSIS (CDID) US ARMY INTELLIGENCE CENTER OF EXCELLENCE 550 CIBEQUE ST BLDG 61730 FT HUACHUCA AZ 85613